



I N T E R W O V E N

# **OpenDeploy® Administration Guide**

## **Release 5.5.1**

© 1999 – 2002 Interwoven, Inc. All rights reserved.

No part of this publication (hardcopy or electronic form) may be reproduced or transmitted, in any form or by any means, electronic, mechanical, photocopying, recording, or otherwise, without the prior written consent of Interwoven. Information in this manual is furnished under license by Interwoven, Inc. and may only be used in accordance with the terms of the license agreement. If this software or documentation directs you to copy materials, you must first have permission from the copyright owner of the materials to avoid violating the law, which could result in damages or other remedies.

Interwoven, TeamSite, OpenDeploy and the logo are registered trademarks of Interwoven, Inc., which may be registered in certain jurisdictions. SmartContext, DataDeploy, Content Express, the tagline and service mark are trademarks of Interwoven, Inc., which may be registered in certain jurisdictions. All other trademarks are owned by their respective owners.

This Interwoven product utilizes third party components under the following copyright with all rights reserved: Copyright 1995-1999, The Apache Group ([www.apache.org](http://www.apache.org)). If you are interested in using these components for other purposes, contact the vendor.



Interwoven, Inc.

803 11th Ave.

Sunnyvale, CA 94089

<http://www.interwoven.com>

Printed in the United States of America

Release 5.5.1

Part # 90-00-20-21-00-551-300

# Table of Contents

## About This Book 11

- Notation Conventions 12
- Other OpenDeploy Documentation 13
  - OpenDeploy Reference 13
  - OpenDeploy Release Notes 13
  - Online Help 13

## Chapter 1: Introduction to OpenDeploy 15

- Task Checklist for Using OpenDeploy 16
- Deployment Planning Considerations 19
  - Case Study: Acme Corp. 20
  - The OpenDeploy Advantage 23
- The OpenDeploy Environment 24
  - Base Server 24
  - Receiver 25
  - Administration Server 25
  - Operations Server 25
  - TeamSite 25
- How OpenDeploy Works 26
  - Source-Target Relationship 26
  - Deployment Configurations 28
  - File Location Definitions 30
  - File Deployment Criteria 31
  - Graphical User Interface 32
- Deployment Types 33
  - Directory Comparison 34
  - TeamSite Comparison 34
  - File List 35
- Deployment Scenarios 37
  - Deployment to a Single Target 37
  - Deployment to Multiple Targets 38
  - Multi-Tiered Deployments 39
  - Transactional Deployments 40
  - Reverse Deployments 41
- Access Rights and Privileges 42

Administrator Role 42

User Role 43

## **Chapter 2: Installation and Configuration 45**

OpenDeploy Software Components 46

Operations Server Software 46

Administration Server Software 47

Base and Receiver Software 48

Software Installation Strategies 49

Single-Host Installation 49

Multi-Host Installation 51

Receive-Only Target Hosts 51

RMI Registry Service Considerations 52

Installation 52

Information Requirements 52

Receiver 54

Installing OpenDeploy into a TeamSite Environment 55

Installing OpenDeploy Software on Windows 56

Installing OpenDeploy Software on UNIX 60

Upgrading From a Previous Release 64

Modifying the Service Configuration File 65

Referencing the Base Server and Receiver Configuration Files 65

Referencing the Nodes Configuration Files 66

Referencing the Bootstrap User Name 66

Configuring the Bootstrap Administrator 67

Defining Target Host Nodes 70

Encoding 71

Logical vs. Fully Qualified Host Names 71

Specifying Host Nodes 71

Modifying the Base Server Configuration File 74

Encoding 76

Communicating with Source and Target Hosts 76

Specifying Downward Revision TeamSite Releases 77

Defining the Scheduler Database 78

Specifying Allowed Hosts for Received Deployments 83

Specifying Allowed Directories for Deployments 84

Logging 85

Encryption 87

Deployment Job Queuing	88
Deploying to OpenDeploy Downward Revision Targets	89
Modifying the Receiver Configuration File	90
Encoding	91
Communicating with Other Hosts	91
Logging	91
File Transport Buffer Size	91
Specifying Allowed Hosts	91
Specifying Allowed Directories	92
Encryption	92
Using Different Administration Server Software	92
Windows	93
UNIX	94
Refreshing the opendeploy.war File	95
Internationalization	96
Service Configuration File Format	96
Encoding for XML-Based Configuration Files	96
OpenDeploy-DataDeploy Integration	97
Component Location	97
Setup	97
Component Descriptions	99
Usage	100
How the Integration Works	101
Performing Authentication Through a Firewall	104
Uninstalling OpenDeploy	105
Windows	105
UNIX	106

## **Chapter 3: Getting Started 109**

Starting OpenDeploy	109
Windows	109
UNIX	111
Starting the User Interface	112
Stopping OpenDeploy	113
Windows	113
UNIX	114
Refreshing the OpenDeploy Server	115
Dependencies on the Operations Server	117

Administration Server	117
Verifying OpenAPI Operation	117
RMI Registry Service Considerations	118
Installing TeamSite in an OpenDeploy Environment	118
OpenDeploy User Interface	119
Browser Refresh Requirements	119
Accessing the User Interface	119
Timeout Setting	121
OpenDeploy Host Management	122
Adding OpenDeploy Hosts	122
Changing Server Information	124
Deleting OpenDeploy Hosts	124
Monitoring Host Logs and Configurations	125
Determining the OpenDeploy Server Version	130
Displaying the OpenDeploy Server Status	131
Roles and Authorization	132
Administrator Role	132
User Role	133
Managing Role Access	133
Assigning User Roles Deployments	138
Composing Deployments	140
Using a Text or XML Editor	140
Using the Deployment Configuration Composer	141
Running Deployments	141
Starting a Deployment	141
Performing a Test Deployment	145
Performing a Simulated Deployment	145
Checking File Integrity on Production Servers	146
Viewing the Source Code of a Deployment Configuration	147
Monitoring Deployments	148
Cancelling Deployments in Progress	153
Logging	155
Log File Location	156
Viewing Log Information	156
Base Server Logging	159
Receiver Logging	161
Macro Deployment Logging	162
Micro Deployment Logging	166

Logging Levels	169
Logging Configuration Settings	171
Logging Rules Hierarchy	173
Log File Size Management	174
Log File Recovery	176
Scheduling	177
Scheduling from the User Interface	178
Scheduling Deployments	179
Viewing Deployment Configuration Schedules	181
Scheduling from the Command Line	185
Uploading Deployment Configurations	192

## **Chapter 4: Deployment Configurations 195**

Deployment Configuration Files	195
Understanding the Configuration DTDs	195
Elements	195
Attributes	197
Encoding	198
Naming Deployment Configurations	199
Deployment Configuration Structure	199
Specifying the Deployment Host	203
Target Replication Farms	203
Definitions	205
Source File Location	206
Target File Location	207
File Filters and Rules	208
Deployment Tasks	208
Transactional	208
Definition-Specific Configurations	209
Directory Comparison Deployments	211
Defining the Source Host File Area	212
Defining the Target Host Area	213
TeamSite Comparison Deployments	214
Defining the Source Host TeamSite Areas	215
Defining the Target Host Location	218
Use With Deploy and Run Scripts	219
File List Deployments	219
Specifying the File List	221

Editing the File List	221
File List Deployments from TeamSite Areas	222
Defining the Target Host Location	223
Defining Source-Based Overrides	223
Defining Target-Based Overrides	225
Mixed Platform Target Areas	225
Specifying Different Target Areas	227
Defining Features on a Target-Specific Basis	227
Deployment Logging	228
Transactional Deployments	229
Use with Multi-Tiered Deployments	230
Fan-Out Deployments	230
Transactional Targets in Fan-Out Deployments	232
Multi-Tiered Deployments	233
Reverse Deployments	236
Using Sample Configurations	238
List Your Target Hosts in the Nodes Configuration File	240
Modify the Target Host to Accept Your Deployment	240
Specify Your Source Host's Name	241
Specify the Target Host's Logical Name	241
Specify the Deployment Type	242
Specifying the Source Host File Locations	243
Specify the Target Host File Location	244
Specify Transactional	245
Modifying the Sample Configuration to Fit Your Needs	245

## **Chapter 5: Advanced Features 249**

Filtered Deployments	249
File System Location-Based Exclusions	249
Pattern-Based Exclusions	251
Configurations	251
Target Host File Deletions Using Filtered Deployments	253
File Comparison Rules	254
File Transfer Rules	256
File Permission Rules	258
User and Group Ownership Transferal	260
Using OpenDeploy with ACLs	262
ACL Names	262

ACE Types	262
Deploying Symbolic Links	264
Source-Side	264
Target-Side	264
Parameter Substitution	265
Deploy and Run	267
Secure Invocation of External Applications on UNIX	267
Requirements	268
Configuration	268
File-Based	269
Directory-Based	270
Deployment-Based	272
Deploy and Run Scripting	273
Deploy and Run Script Logging	275
Communicating Status to OpenDeploy	278
Deploying to a Package File Using Deploy and Run	278
Encryption	280
Symmetric Key Encryption	280
Asymmetric Key Encryption	282
Redeploying Legacy Web Sites	291

## **Chapter 6: Composing Deployments 293**

Deployment Configuration Composer	293
Tree and Errors Tabs	294
Details Pane	295
Types of Deployment Configuration Settings	296
Global Deployment Settings	296
Definition Settings	297
Creating a New Configuration	298
Naming the Deployment Configuration	298
Specifying the Log Rules	299
Verifying or Changing the Source Host Name	300
Specifying Deployment Encryption	301
Naming the Replication Farm Element	303
Adding Target Host Nodes to the Replication Farm Element	303
Assigning Next-Tier Deployments to Target Hosts	304
Specifying a Transactional Deployment.	305
Enabling Deployments to Downward Revision Hosts	305

Setting Deploy and Run	306
Naming and Adding Definitions	315
Selecting the Definition Type	315
Defining the Source File Location	316
Defining a Subdirectory Within the Source File Location	320
Applying Source-Side Filters	321
Following Source-Side Symbolic Links in Deployments	323
Designating Alternate Targets from the Source	324
Defining the Target File Location	325
Applying Comparison Rules	326
Applying Transfer Rules	328
Applying Permission Rules	329
Applying Target-Side Filters	334
Saving the Deployment	335
Editing Deployment Configurations	336

## **Glossary 337**

## **Index 345**

# About This Book

---

*OpenDeploy Administration Guide* is a guide to install, configure, and use OpenDeploy®. It is primarily intended for webmasters, system administrators, and those involved in deploying content between development servers and production servers.

If you are using OpenDeploy in conjunction with TeamSite®, you should also know TeamSite functionality and terminology. Many of the operations described in this manual require *root* or *Administrator* access to the OpenDeploy host server. If you do not have root or Administrator access to the OpenDeploy host server, consult your system administrator.

This manual uses the term “Windows” to indicate any supported version of the Microsoft Windows operating system, such as Windows NT® or Windows® 2000.

This manual uses the term “UNIX” to indicate any supported flavor of the UNIX® operating system.

Windows: Users should be familiar with either IIS or Netscape® Web servers, and with basic Windows server operations such as adding users and modifying Access Control Lists (ACLs).

UNIX: Users of this manual should be familiar with basic UNIX commands and be able to use an editor such as emacs or vi.

It is also helpful to be familiar with regular expression syntax. If you are not familiar with regular expressions, consult a reference manual such as *Mastering Regular Expressions* by Jeffrey Friedl.

## Notation Conventions

This manual uses the following notation conventions:

Convention	Definition and Usage
<b>Bold</b>	Text that appears in a GUI element (for example, a menu item, button, or element of a dialog box) and command names are shown in bold. For example:  Click <b>Edit File</b> in the Button Bar.
<i>Italic</i>	Book titles appear in italics. Terms are italicized the first time they are introduced. Important information may be italicized for emphasis.
Monospace	Commands, command-line output, and file names are in monospace type. For example:  The <code>iwodstart</code> command-line tool starts an OpenDeploy deployment task.
<i>Monospaced italic</i>	Monospaced italics are used for command-line variables. For example:  <code>iwodstart deployment</code>  This means that you must replace <i>deployment</i> with your values.
<b>Monospaced bold</b>	Monospaced bold represents information you enter in response to system prompts. The character that appears before a line of user input represents the command prompt, and should not be typed. For example:  <b><code>iwodstart</code></b>
<b><i>Monospaced bold italic</i></b>	Monospaced bold italic text is used to indicate a variable in user input. For example:  <b><i><code>iwodstart deployment</code></i></b>  means that you must insert the values of <i>deployment</i> when you enter this command.
[ ]	Square brackets surrounding a command-line argument mean that the argument is optional.

Convention	Definition and Usage
	Vertical bars separating command-line arguments mean that only one of the arguments can be used.

## Other OpenDeploy Documentation

In addition to this Administration Guide, OpenDeploy includes the following documentation components:

- *OpenDeploy Reference*
- *OpenDeploy Release Notes*
- Online help

### OpenDeploy Reference

*OpenDeploy Reference* is a manual that contains reference material on OpenDeploy configuration DTDs and command-line tools (CLTs). Use this information to find information on a specific DTD or CLT quickly and easily.

### OpenDeploy Release Notes

*OpenDeploy Release Notes* contains supplemental and late-breaking information regarding OpenDeploy not found in the other documentation. Refer to the OpenDeploy Release Notes for information regarding supported platforms, installation requirements, new features and enhancements, and known issues.

### Online Help

OpenDeploy includes online help topics associated with each window in the OpenDeploy user interface. You can access the associated help topic by clicking the Help link in the upper right-hand corner of the window. The help topic will appear in a separate browser window that you can move and resize. You can display a navigation panel that provides you access to all the help topics by clicking the Show Navpane button. This panel provides you the ability to access help topics through the contents, index, and by using keyword searching.



## Chapter 1

# Introduction to OpenDeploy

---

Welcome, and congratulations on selecting the Interwoven OpenDeploy software! OpenDeploy is an industry-leading content distribution product for deploying static and dynamic Web content to a multi-tier, multiple server environment. OpenDeploy runs on a variety of common servers, and is well-suited for a cross-platform enterprise.

OpenDeploy is a key part of your enterprise Web content management infrastructure, along with software products such as TeamSite, TeamSite Templating, DataDeploy™, and OpenSyndicate™. Using OpenDeploy in conjunction with these products provides you added features and flexibility otherwise not available. However, you can also elect to use OpenDeploy separately from such products for moving your Web content files.

OpenDeploy is based on advanced technology providing Web site content transfer capability based on the following principles:

- **Performance** — OpenDeploy has significantly improved the transaction speed for large deployments, such as those with tens of thousands of files, over previous releases.
- **Scalability** — You can add additional hosts easily. Your ability to deploy content to multiple production servers simultaneously is greatly enhanced.
- **Ease of use** — A browser-based user interface allows easy configuration and use. Installation is also simplified, as no client software installation is required for you to operate OpenDeploy remotely from the actual OpenDeploy server. All you need to access OpenDeploy is a supported browser.
- **Transactional capability** — OpenDeploy supports transactional deployments, which will restore a server host receiving deployed files back to its previous existing state in the event the deployment is unsuccessful.
- **Security** — Defined user and administrator deployment roles ensure that individuals can only perform the tasks for which they are authorized. Encrypted deployments protect against unauthorized access to files.

- Flexibility — OpenDeploy supports the use of external scripts through its Deploy and Run feature.

## Task Checklist for Using OpenDeploy

This section provides a checklist of key tasks you must perform to install, configure, and use OpenDeploy.

- Read this chapter in its entirety before proceeding to the next step. Here you will find an overview of the OpenDeploy product, how it works, and how you can best use it to meet your organizational needs.
- Install the OpenDeploy software components on a single or multiple server hosts. See “Installation” on page 52 for more information.
- Configure your bootstrap administrator. This is required for you to log on to OpenDeploy for the first time. See “Configuring the Bootstrap Administrator” on page 67.
- Define the targets in your host’s nodes configuration file. Each target host must have the base server or receiver software installed on it. See “Defining Target Host Nodes” on page 70.
- Modify your OpenDeploy host’s base server configuration file to allow incoming deployments from other OpenDeploy hosts. You can also configure settings regarding your host’s logging, scheduling, encryption, and other features. See “Modifying the Base Server Configuration File” on page 74.
- Modify the receiver configuration files of any target hosts that have the receiver software installed on them. See “Modifying the Receiver Configuration File” on page 90.
- Start the OpenDeploy services or daemons, which provide the functionality of the product. See “Starting OpenDeploy” on page 109.
- Access the OpenDeploy user interface. See “OpenDeploy User Interface” on page 119.
- Add the OpenDeploy hosts that you want to view and use to the user interface. See “OpenDeploy Host Management” on page 122.

- Add Administrator and User roles to those who will administer OpenDeploy host servers and run specific deployments on them. See “Roles and Authorization” on page 132.
- Familiarize yourself with how to start a deployment. See “Starting a Deployment” on page 141.
- Perform and monitor a test deployment on your OpenDeploy host using the default configuration settings. See “Performing a Test Deployment” on page 145.
- Familiarize yourself with how to perform a simulated deployment. This feature is useful for seeing how a deployment would take place without actually moving the files. You can also use it to determine if files on the target host are different from those on its source host, which can be indicative of accidental or intentional modification of Web site files. See “Performing a Simulated Deployment” on page 145.
- Familiarize yourself with OpenDeploy logging. See “Logging” on page 155.
- Familiarize yourself with OpenDeploy deployment scheduling. See “Scheduling” on page 177.
- Familiarize yourself with how OpenDeploy deployment configurations work and are structured. Although you can create and edit deployment configuration files using the Deployment Configuration Composer, you should also know how the configuration files themselves are structured, and how to make changes in the files using a text or XML editor. See “Deployment Configuration Files” on page 195.
- Modify a sample deployment configuration to work on your OpenDeploy host and perform a practical deployment of files. See “Using Sample Configurations” on page 238.
- Familiarize yourself with the OpenDeploy deployment types:
  - Directory comparison, where files on a source and target file system directory are compared and the differences are deployed. See “Directory Comparison Deployments” on page 211.
  - TeamSite comparison, where files on two TeamSite areas are compared and the differences are deployed. See “TeamSite Comparison Deployments” on page 214.
  - File list, where a list of files on the source list determines which files are deployed. See “File List Deployments” on page 219.
- Familiarize yourself with how to define target host-based overrides of deployment configuration settings. This allows you to modify how files are deployed on a target-by-target basis. This is useful when deploying files to multiple targets. See “Defining Target-Based Overrides” on page 225.



- Familiarize yourself with specialized deployments, such as:
  - Transactional deployments, which will roll back a deployment and restore the target hosts to their previous states if a deployment is unsuccessful for any reason. See “Transactional Deployments” on page 229.
  - Fan-out deployments, which deploy a set of files to several target hosts simultaneously. See “Fan-Out Deployments” on page 230.
  - Multi-tiered deployments, which allow a deployment to be deployed across a series of OpenDeploy sender hosts and their respective target hosts. See “Multi-Tiered Deployments” on page 233.
  - Reverse deployments, which allow files that are added or modified on a target host to be deployed back to the sending host. See “Reverse Deployments” on page 236.

OpenDeploy includes sample files of these deployments that you can review and modify for your own use. See “Using Sample Configurations” on page 238 for more information.

- Familiarize yourself with other OpenDeploy features as desired. See “Advanced Features” on page 249.
- Familiarize yourself with how to compose and edit deployments using the Deployment Configuration Composer. This is a tool in the OpenDeploy user interface that allows you to create new or edit existing deployment configurations more easily and with less risk of error than by modifying the deployment configuration file using a text or XML editor. See “Composing Deployments” on page 293.

## Deployment Planning Considerations

The following section poses questions regarding common planning issues. It then describes how OpenDeploy can meet those issues.

### **How is content being distributed in your company?**

Is there a standardized method for moving Web files from the development server or hub to the production servers. There are a variety of open and proprietary products for doing this task, but they have varying levels of reliability and support. Additionally, many are platform-specific and are ill-suited for a large enterprise with a combination of Windows and UNIX servers.

### **Are you distributing the content manually?**

In many enterprises, the task of moving Web files from their development servers to their destinations requires an employee to manually move the files. There is little or no automation, and the risk of user error exists each time files are moved. There is also no scheduling capability, requiring the deployment administrator to be present when the deployment occurs.

### **Do you have more than one Web server or multiple data centers?**

Larger enterprises often have multiple Web servers and data centers. Some might be mirror sites requiring each server or data center to store the exact same files. In other cases, multiple data centers might have a portion of their Web sites customized to their particular audiences, for example, different geographical regions.

### **How do you synchronize multiple servers with new or updated content?**

An enterprise with multiple data centers or mirrored sites needs the ability to synchronize its Web servers to contain the same content. A high level of precision might also be required, such as if a production Web server were being updated at a specific time to coincide with a product release.

## Do you have any legal compliance that requires you to prove what content is on your Web Site on a particular day?

As the Web is becoming an increasingly bigger component of an enterprise's business model, legal compliance of Web site content is required by many companies that conduct business on the Internet. An enterprise must be able to create and archive a snapshot of their Web sites at a particular point of time, and even be able to "roll back" their production Web servers to an earlier version.

### Case Study: Acme Corp.

In this section we introduce a fictional company named *Acme*, headquartered in San Francisco with regional data centers in New York, London, and Tokyo. There are a mix of Windows and UNIX development and productions Web servers. Recently, Acme adopted TeamSite as its Web content management solution, and is contemplating adding OpenDeploy as well.

Currently, each regional data center is responsible for receiving Web content from the home office in San Francisco, making any additions or changes, and moving the files to their regional production servers. Because of rapid growth and the acquisition of smaller companies, no single deployment solution exists. Instead, different groups within the enterprise use their own individual deployment solutions. Here is how each regional component moves files:

- San Francisco — locally scripted implementation of FTP
- New York — third-party deployment software with minimum features
- London — Rsync
- Tokyo — deployment software for UNIX platforms only

Each of these solutions has limitations in the areas of support, cross-platform compatibility, and scalability. Additionally, each solutions requires a separate knowledge base and level of expertise. If Acme could standardize on a single deployment solution that met the needs of each regional component, but also ensured reliable support and a common knowledge base, deployment of Web content would be much easier to execute and manage.

Currently, deployments are performed by a small number of administrators with a keen understanding of how the deployment software works. In some cases, there is no scheduling capability, requiring one of these administrators to be present whenever a deployment takes place. Deployments typically occur after regular work hours to avoid slowing the network with a high level of traffic. As a result, the deployment administrator must work non-standard hours to oversee the deployment. If no administrator is available, the deployment cannot take place.

Many of the deployments send the same content to multiple sites. For the existing solutions with no multiple target capability, the same deployment procedures must be performed serially for each recipient server. This is not only a time-consuming and tedious task, but it increases the risk of a user or system error that can compromise the deployment. If the recipient servers include one or more mirror sites of a production server, it is vitally important that all of these servers contain the exact same content. It is preferable that the deployment not go out at all if it is not guaranteed that all sites will receive the new files simultaneously.

The amount and size of the Web files themselves can vary. Acme's Web sites are updated frequently. In some cases, the update requires the addition of only a few files. In other cases, the entire Web site needs updating, requiring the deployment of thousands of files. Ideally, Acme's deployment solution should be able to determine what files need to be deployed for each situation, rather than simply redeploying an entire set of Web site files every time.

Based on this scenario, Acme needs the following features and capabilities from its deployment solution:

- A single solution that can be used by all the data centers — a single solution greatly simplifies issues of upgrades, support, and training.
- Full software support and training — reduce the pressure on the Acme staff to teach themselves the product and attempt to support or troubleshoot it.
- Cross-platform capability — require the software to not only run on different platforms, but also to be able to send and receive files from servers of differing platforms.

- Multi-target deployment capability — being able to send a deployment to multiple targets simultaneously speeds and simplifies transferring data from development servers and data center hubs to their destinations. User and system errors are also reduced.
- Multi-tiered deployment capability — allow multiple OpenDeploy servers to redeploy files to multiple tiers of servers. This feature allows the “chaining” of deployments from one tier to the next. For example, a file set could be deployed from the development server to a data center, and then redeployed to multiple production servers automatically.
- Scheduling of deployments — allow deployments to take place during periods of lower network traffic, such as after work hours or on the weekend. Deployments can be scheduled to coincide with other activities, such as a new project launch. Employees do not have to be present during the deployments, freeing them from working odd hours and allowing them to concentrate on other activities.
- Determination of deployable files — be able to identify the delta between a like set of files on the source and target hosts. This process identifies only those files that require deployment when updating a Web site, avoiding deploying the identical or unnecessary files.
- Mobility of deployment administration — be able to administer and monitor deployments from multiple locations throughout the enterprise remotely.
- TeamSite integration — tightly integrate the deployment solution with the features and capabilities of TeamSite, including deploying from TeamSite areas to file system locations on the target hosts.

## The OpenDeploy Advantage

OpenDeploy can meet Acme's requirements in the following ways:

- OpenDeploy is an established product with full documentation, technical support, and training to back it up. Upgrades are regularly made available, either on an individual or subscription basis.
- OpenDeploy provides an automated solution based on customizable configurations. After you set up your deployment configurations, you can schedule them to occur whenever you want, without an administrator present.
- Each deployment follows the exact specifications of the configuration, eliminating the possibility of user error when moving files. You also can relieve employees from having to move files manually, and redirect their efforts to other tasks.
- OpenDeploy can perform a comparison of the files residing on the development server and the recipient targets, or between two TeamSite areas. Only those files that meet the deployment criteria are deployed, saving time and network resources.
- OpenDeploy allows you to deploy a set of files from a single source host to any number of recipient hosts. If you have multiple tiers of servers, for example development servers, hubs, and production servers, you can use multiple OpenDeploy servers to redeploy files from one tier to the next.
- OpenDeploy provides IT managers a high level of flexibility in assigning and managing administrative and user role access to deployments. Specific individuals can have access to starting and scheduling specific deployments based on departmental needs without allowing undo access or burdening IT staff.
- OpenDeploy uses a browser-based interface accessible from any computer within the enterprise for many deployment and monitoring tasks, allowing deployment administrators to start and monitor deployments from remote locations.
- OpenDeploy can deploy from TeamSite staging areas, editions, and workareas to file system locations on the target hosts. You can redeploy legacy TeamSite editions in case you want to "roll back" a Web site to a previous version or recreate a past Web site.

## The OpenDeploy Environment

OpenDeploy consists of a suite of interlocking services that create the OpenDeploy environment. Within the OpenDeploy environment are the following components:

- Base server software — enables the host to start deployments to other hosts, as well as to receive files deployed from other OpenDeploy hosts.
- Receiver software — enables the host only to receive deployed files.
- Administration server software — manages and generates the user interface.
- Operations server software — manages the administrator and user accounts.
- TeamSite (optional) — OpenDeploy can deploy files from selected TeamSite staging areas, editions, and workareas to file system locations on the target hosts.

### Base Server

The core of the OpenDeploy environment is the *base server*. A base server is a host with the base server software installed and configured on it. A base server can both send and receive deployed files. When the base server deploys files to another host, it assumes the role of the *source host* in the source/target relationship. If the base server itself is receiving deployed files, it becomes the *target host*.

The base server can be a development server within the enterprise firewall, or it might be a hub outside the firewall responsible for redeploying files it received to another set of target hosts. The number and positioning of base servers in the OpenDeploy environment is determined by the deployment requirements.

The base server host is where the *deployment configurations* are located. Deployment configurations are XML-based files that determine how and to where a deployment will take place. The base server also maintains a variety of log files for each deployment and for general base server activities.

## Receiver

A *receiver* is a host with the receiver software installed on it. A receiver host can only receive files as a target host in the source/target relationship. Receiver hosts are typically production servers outside the firewall that serve the deployed content to its intended audience but do not need to redeploy the content files any further.

## Administration Server

The OpenDeploy administration server is responsible for managing and generating the browser-based OpenDeploy user interface. The administration server does not send or receive files itself, but works in conjunction with the source and target hosts. The administration server software can reside with all the other OpenDeploy components on a single server, or it can reside on a host separate from the base server or receiver software. However, in all cases, it must be co-located with the operations server.

## Operations Server

The OpenDeploy operations server manages the Administrator and User role access to OpenDeploy, including the ability to create and start deployments. Like the administration server, the operations server does not send or receive files, but instead provides the access management infrastructure for all the administrators and users within the OpenDeploy environment. The operations server software can reside with all the other OpenDeploy components on a single server, or it can reside on a host separate from the base server or receiver software. However, in all cases, it must be co-located with the administration server.

If you are using OpenDeploy in conjunction with TeamSite 5.0 or later, you must use the TeamSite OpenAPI instead of the operations server software.

## TeamSite

OpenDeploy can be tightly integrated with TeamSite, allowing you to manage and deploy your content within the same environment. In many cases, customers install their TeamSite and OpenDeploy base server software on the same server host. OpenDeploy can compare two TeamSite areas, such as staging areas, editions, and workareas, and deploy the differences to a file system location on the target host. OpenDeploy can compare two TeamSite areas and deploy only those files that are different.

## How OpenDeploy Works

The OpenDeploy source host processes deployment configurations. These configurations determine the type of deployment being performed, as well as what other functions and features OpenDeploy will perform in the course of the deployment. The deployment configuration also specifies what target hosts will receive the deployed files.

### Source-Target Relationship

A deployment of content files begins at the *source host* (the host sending the files) and ends at the *target host* (the host receiving the files).

Depending on the software you install on a server, an OpenDeploy host can act as both a source and target host, or as a target host only. Source hosts are not restricted in the number of target hosts available to them, providing the target host has the proper software installed and licensed. Similarly, a target host can receive deployments from any number of source hosts.

In most enterprises, a single source host will deploy files to many target hosts. Depending on the type of deployment taking place, the deployed files will either be exactly the same for all targets, or customized based on a comparison of file differences between the source host and each target host.

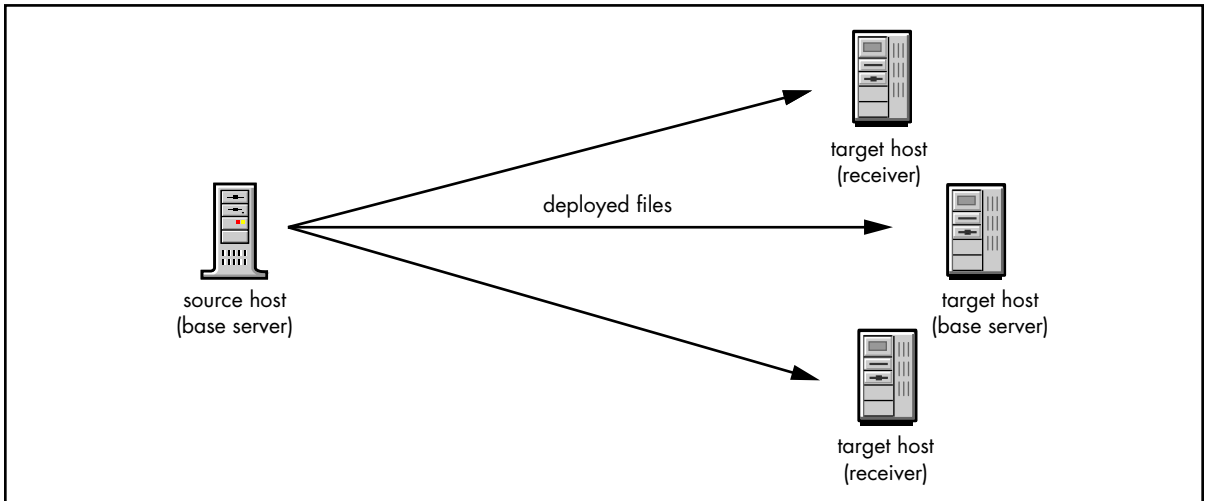
#### Source Host

The source host is a server with the OpenDeploy *base server* software installed. The source host originates and manages all deployments. The source host maintains a list of all target hosts to which it can deploy files. You can deploy files from one or more source file locations, which can be either file system locations or TeamSite areas.

#### Target Host

The target host is a server with the OpenDeploy base server or receiver software installed. Any such server in the OpenDeploy environment can receive deployed files, as long as the server is known to the source host and has declared itself available for receiving deployments from the appropriate source hosts.

Figure 1 shows the source/target relationship between a source host (with the base server software installed) and three target hosts (one with the base server software and two with the receiver software installed).



*Figure 1: Single Base Server and Multiple Receivers*

Those target hosts that have the base server software installed are capable of redeploying the files it receives to another set of targets. A deployment originating from a single source host might be redeployed several times to more and more targets automatically, saving time and labor, and ensuring content synchronicity among all the OpenDeploy hosts. In this case, the host receiving the original deployment is a target host, but when it redeploys those files to a new set of target hosts, it does so as a source host. This server must have the appropriate software installed and licensed to be able both to receive and redeploy files.

In Figure 2, one of the target hosts for the first deployment has the base server software installed, and can redeploy as a source host the files it received as a target host.

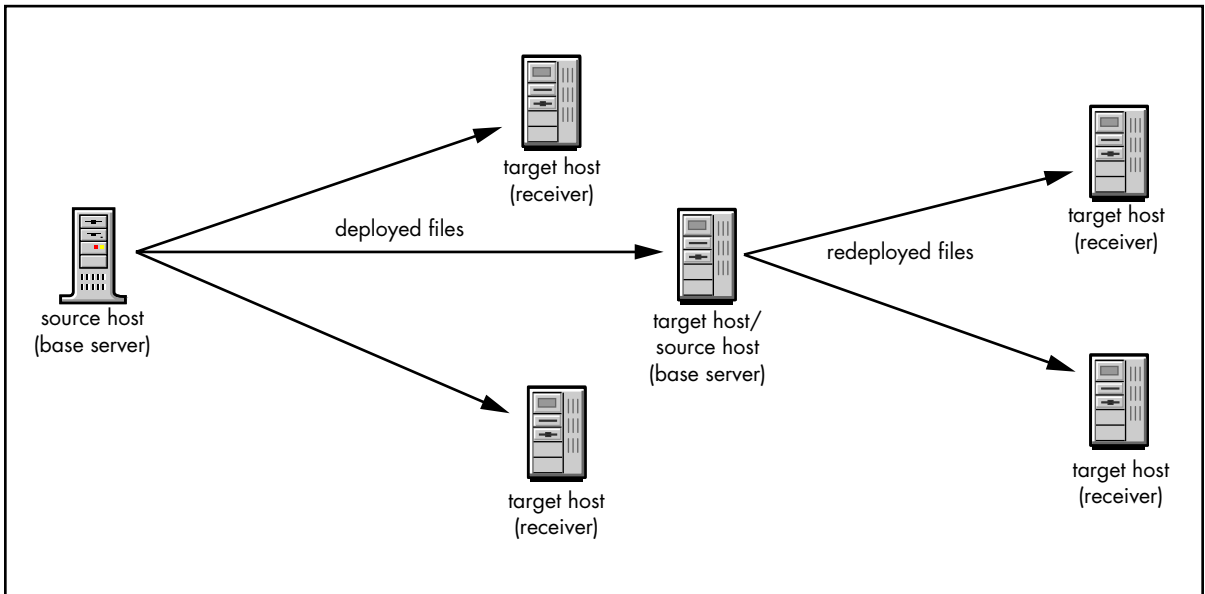


Figure 2: Redeploying Files Using Multiple Source Hosts

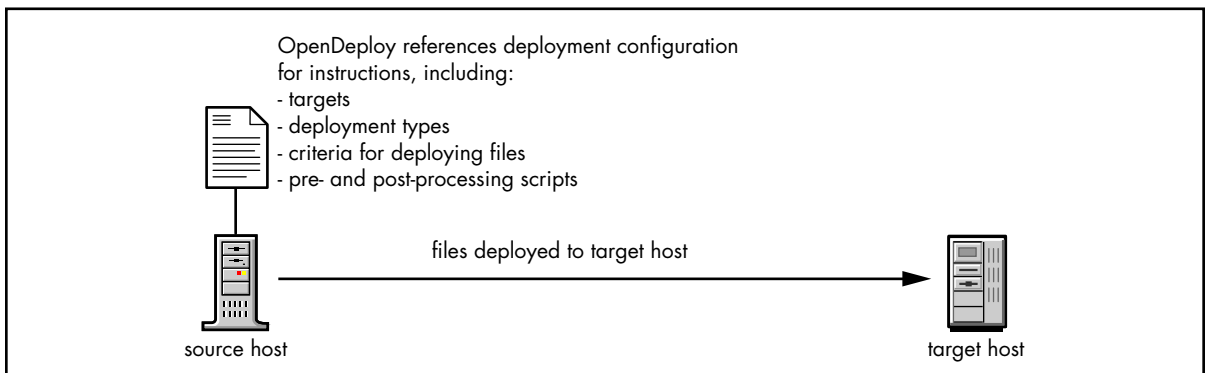
## Deployment Configurations

The criteria for how to determine which files get deployed is specified within a *deployment configuration*. An OpenDeploy source host can process any number of distinctly named deployment configuration files, each of which can deploy files to different target area on different target hosts under different conditions. A typical deployment configuration will specify the following:

- Target host receiving the deployment
- Source file locations
- Target hosts
- Target file location
- Type of deployment

- Logging
- Use of external pre- and post-processing scripts
- Filters for limiting deployed files
- Rules for comparing files
- Rules for transmitting files
- Rules for setting file and directory permissions on deployed files

Figure 3 shows how a source host uses the information in a deployment configuration file to perform a deployment.



*Figure 3: How OpenDeploy Uses Deployment Configurations*

## File Location Definitions

From the source host, one or more source file locations can be specified to deploy its files to a single target file location. The grouping of one or more source file locations deploying files to a target is known in the deployment configuration as a *definition*, and is the basis of any deployment. Each definition must have a unique name which you can assign to it, either by editing the deployment configuration file, or using the Deployment Configuration Composer.

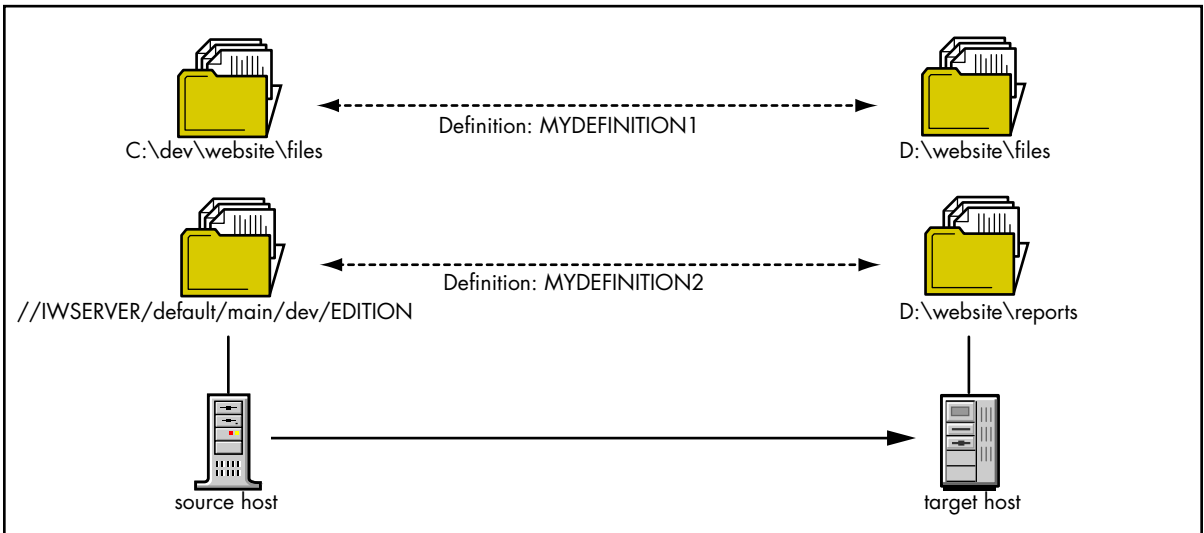


Figure 4: Definitions

Figure 4 shows how a single OpenDeploy host can deploy files from multiple source file locations (file system locations or TeamSite areas) to multiple target file locations (file system locations only) on the same source and target host. Each definition can have one or more source file locations, but only a single target file location.

## File Deployment Criteria

OpenDeploy determines file *deployment criteria*—whether a file or directory should be deployed from the source host to the target host—based on one of the following methods:

- Comparing files between file system locations on a source and target host and determining the differences
- Comparing two TeamSite areas on the source host with TeamSite installed and determining the differences
- Deploying files referenced in a list of files without comparing them

### Comparison-Based Deployment Eligibility

A comparison-based deployment is the result of comparing one set of files with a second set of the same files. These file sets can be either file system locations on the source or target hosts, or a pair of TeamSite areas within the same backing store on a source host that has TeamSite installed. You can configure OpenDeploy to compare two TeamSite areas in any single backing store on a multi-backing store TeamSite host.

The default comparison criteria used to determine whether or not a given file should be deployed are:

- Modification date (when the source-side file is newer than its target-side equivalent)
- Type mismatch (a file and a directory sharing the same name)
- Size difference

In addition, OpenDeploy provides a variety of other deployment criteria you can use or ignore in your deployment configurations, including:

- Source-side file is older than target-side equivalent
- Access control list (ACL) difference (Windows only)
- User difference (UNIX only)
- Group difference (UNIX only)
- Permission difference (UNIX only)

See Chapter 5, “Advanced Features” on page 249 for a complete description of all comparison-based deployability methods and criteria.

## File List-Based Deployment Criteria

File list deployments do not compare files or directories, but simply deploy the files from the source host based on a specified list of files. This list of files resides on the source host and is referenced in the deployment configuration.

## Graphical User Interface

OpenDeploy provides a browser-based user interface with which you can create, schedule, start, and monitor deployments. In many cases, you will find the user interface to be sufficient to meet your OpenDeploy needs. However, certain more advanced OpenDeploy configurations require you to modify the XML-based elements and attributes of the deployment configuration files.

Figure 5 shows an example of the browser-based user interface. The navigation pane on the left presents you with full access to all the major task and monitoring windows. The details pane on the right displays the contents of the item in the navigation pane you selected.

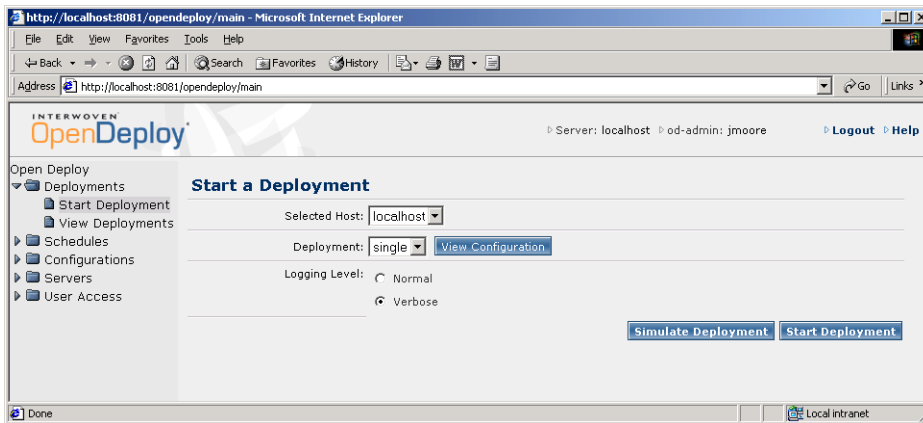


Figure 5: Example of the OpenDeploy User Interface

Here are some of the tasks you can perform using the OpenDeploy user interface:

- View the configuration file information for a selected deployment.
- Create a new deployment and edit the configuration of an existing one.
- Schedule deployments, either on a one-time basis, or recurrent by minute, hour, day, week, or month.
- Start a deployment manually.
- Monitor deployments, including logging, pending status, and success or failure.

## Deployment Types

OpenDeploy uses one of the following methods for deploying files from the source host to the target hosts:

- Directory comparison
- TeamSite comparison
- File list

The following sections describe each of them in detail, including when each one is best suited as your deployment choice.

## Directory Comparison

Directory comparison is the default deployment type used by OpenDeploy. Directory comparison deployments compare a specified directory on the OpenDeploy server sending the files (the source host) and a corresponding server receiving the deployed files (the target host). New or updated files on the source host are then copied to the target host. After the deployment is complete, the source and target hosts' files are the same. See “File Deployment Criteria” on page 31 for information on how OpenDeploy determines which files and directories are eligible to be deployed.

You can include multiple file system locations in your deployment. The files residing in each of these specified file system locations will be compared with the target files, and the deployable files sent to the target host.

If OpenDeploy is performing a fan-out deployment, it will compare the source host's files with each of the target hosts' files. The files deployed will reflect the differences between the source host and each respective target host. That way, even if each of the target hosts have a different set of files, following the deployment, they will all be in sync with the source host and with each other.

Directory comparison deployments provide a full synchronization of the content between the source and target hosts. This is the most comprehensive way to ensure that the content in the source and target hosts are identical. Directory comparison is also the most resource-demanding method for time and network bandwidth because file and directory information must be exchanged over the network to determine the source and target hosts' file differences.

## TeamSite Comparison

TeamSite comparison compares two TeamSite areas within the same backing store on the source host with TeamSite installed, and deploys the differences to the target hosts. You can configure OpenDeploy to compare two TeamSite areas in any single backing store on a multi-backing store TeamSite host. The source host must have TeamSite installed and configured to use this type of deployment. The TeamSite workareas, staging areas, and editions all can be specified for a TeamSite comparison deployment. See “File Deployment Criteria” on page 31 for information on how OpenDeploy determines which files and directories are eligible to be deployed.

In a TeamSite comparison, you specify the TeamSite area containing the updated files that you want to deploy. This area is typically the branch staging area or an edition. You must also specify another TeamSite area that contains a mirror image of the files on the target hosts, typically a previously created edition. TeamSite will compare the area containing the updated files with the area containing the existing target host files and deploy the differences to the specified target hosts.

You can also simply configure OpenDeploy to determine the latest and next-to-latest TeamSite editions automatically, and deploy the differences between them. This is a common method of integrating OpenDeploy and TeamSite.

Unlike a directory comparison deployment, the TeamSite comparison takes place solely on the source host. OpenDeploy assumes that the files in the previous area are identical to those on the target hosts themselves. The deployed files are moved to the file system location on the target host specified in the deployment configuration.

Because the TeamSite comparison is done purely on the TeamSite source host, it is faster and less bandwidth-intensive than the directory comparison. No network traffic is generated before the file deployment itself occurs. However, TeamSite comparison is totally dependent on the source host having a perfect snapshot of the files residing on the target host. If files have been changed on the target host, it is up to you to ensure that the corresponding TeamSite area on the source host is updated as well. OpenDeploy can reverse-deploy from a target host to the source host, but it cannot independently determine when that will be necessary.

## File List

A file list-based deployment does not compare files on the source and target hosts or between TeamSite areas. Instead, OpenDeploy moves files based on a predetermined list specifying the files to be deployed. The files and their paths in the list are in locations relative to the file system-based area specified in the deployment configuration. This list can be a fixed or static file, or it can be generated dynamically using some generation tool such as the TeamSite `iwevents` command-line tool. Refer to your TeamSite documentation for information on TeamSite command-line tools.



The entries in the file list are influenced by the file system syntax of the host server. Forward slashes (“/”) can be used for either Windows or UNIX hosts:

```
www/index.html  
www/andre/index.html  
www/products.html
```

while backslashes (“\”) are only permitted on Windows hosts:

```
www\index.html  
www\andre\index.html  
www\products.html
```

In these examples, `www` is a directory immediately subordinate to the area location on the source host as specified in the deployment configuration. For example, if the area specified for a source host is the following directory:

```
C:\webfiles
```

then the entries in the file list example combined with the specified area would end up being:

```
C:\webfiles\www\index.html  
C:\webfiles\www\andre\index.html  
C:\webfiles\www\products.html
```

A file list deployment is simpler and more predictable than the file system and TeamSite comparison deployments previously described. If a large number of smaller files are involved, a file list deployment can take less of a toll on network resources than a file system comparison, and can process the deployment faster than a TeamSite comparison. However, if very large files are involved, deploying all the files specified in the file list might be less efficient than a comparison-based deployment where only those files that have changed are deployed.

## Deployment Scenarios

OpenDeploy allows a wide range of deployment scenarios to meet every possible need. The following sections describe the various methods with which you can deploy files between OpenDeploy hosts.

### Deployment to a Single Target

The simplest deployment of files is from the source host to a single target host. In Figure 6, the OpenDeploy source host references the deployment configuration for a single target deployment. This configuration specifies a forward deployment to the target host based on one of the three types of deployments:

- Directory comparison
- TeamSite comparison
- File list

The source host subsequently deploys those files to the target host, and the deployment is completed successfully.

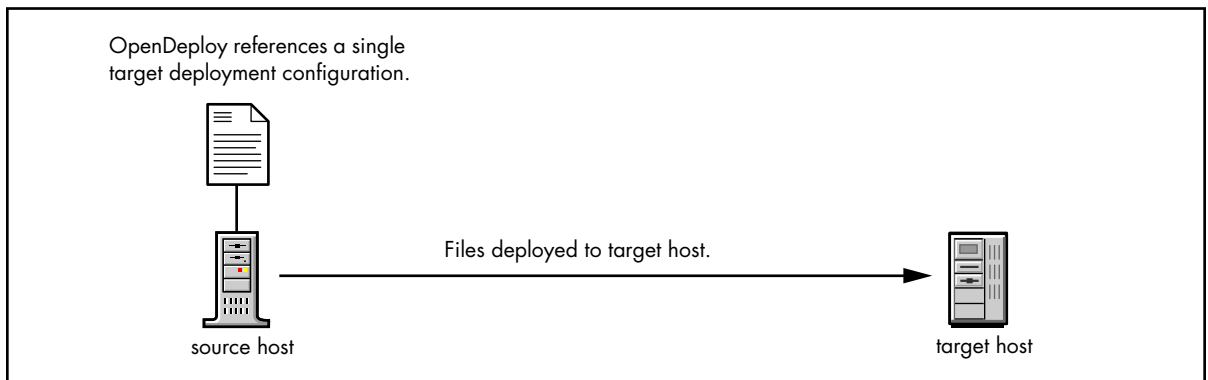


Figure 6: Deployment to a Single Target Host

## Deployment to Multiple Targets

This deployment configuration specifies that a set of files should be deployed to two or more target hosts. In OpenDeploy, this is referred to as a *fan-out deployment*.

In Figure 7, the OpenDeploy source host server *A* references the deployment configuration for a fan-out deployment. This configuration specifies a fan-out deployment to the target hosts *A-1*, *A-2*, and *A-3*.

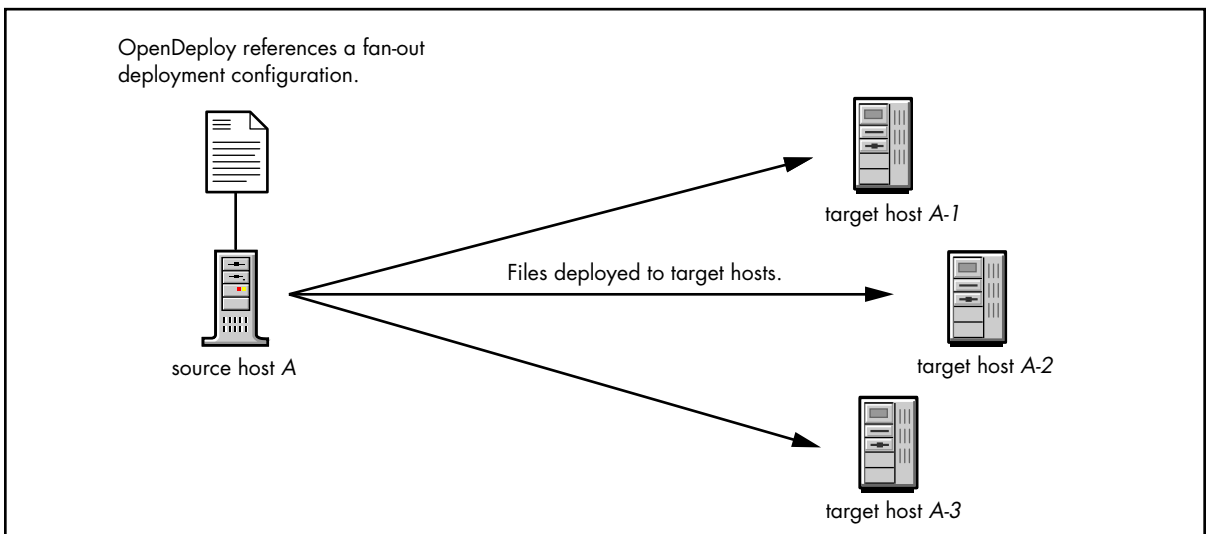


Figure 7: Deployment to Multiple Target Hosts

## Multi-Tiered Deployments

A *multi-tiered deployment* employs one or more target OpenDeploy servers to deploy files to another set of servers, known as a *tier*. Each source host and its target hosts represent a separate tier. Like fan-out deployments, multi-tiered deployments allow the automatic deployment of files to a wide range of recipient targets in your enterprise with little more effort than deploying to a single target host.

In Figure 8, the OpenDeploy source host *mars* performs a fan-out deployment as in Figure 7. Of the three target hosts receiving the deployed files, *venus* is an OpenDeploy host with the base server software installed and is capable of performing deployments. Like *mars*, *venus* also references its own specified deployment configuration for its own deployment of files to *mercury* and *neptune*.

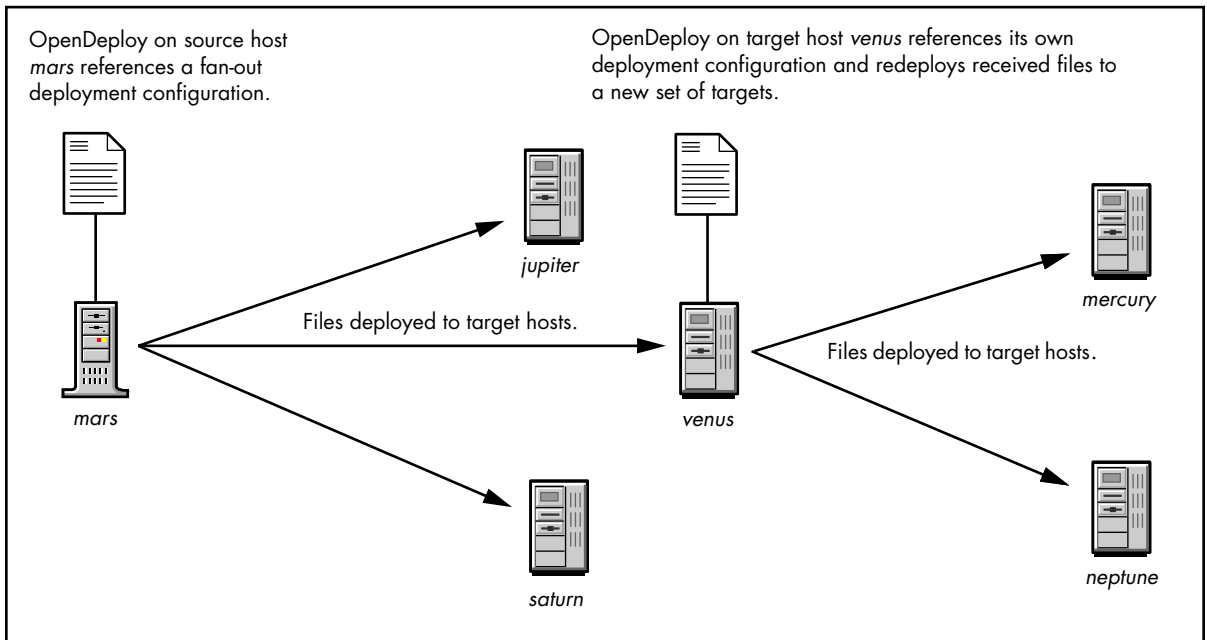


Figure 8: Multi-Tiered Deployment

## Transactional Deployments

If one or more targets in a deployment fail to successfully receive the deployed files, you can configure OpenDeploy to automatically roll back the deployment that took place and restore all the target hosts to their previous states. This feature is known as a *transactional deployment*. This ability is vital if you have multiple production servers that must perfectly mirror each other. Transactional deployments ensure that disparities between production Web servers will not exist as the result of a failed or problematic deployment to multiple targets. In the case of multi-tiered deployments, a transactional deployment will roll back the deployment at that tier rather than the entire multi-tiered deployment.

In Figure 9, *mars* has started a deployment with the transactional feature enabled to *venus*. This deployment has subsequently failed. Upon determination that the deployment failed, OpenDeploy removes the portion of the deployment that did make it to *venus*, and restores that host's files to its previous state, effecting a rollback of the failed deployment.

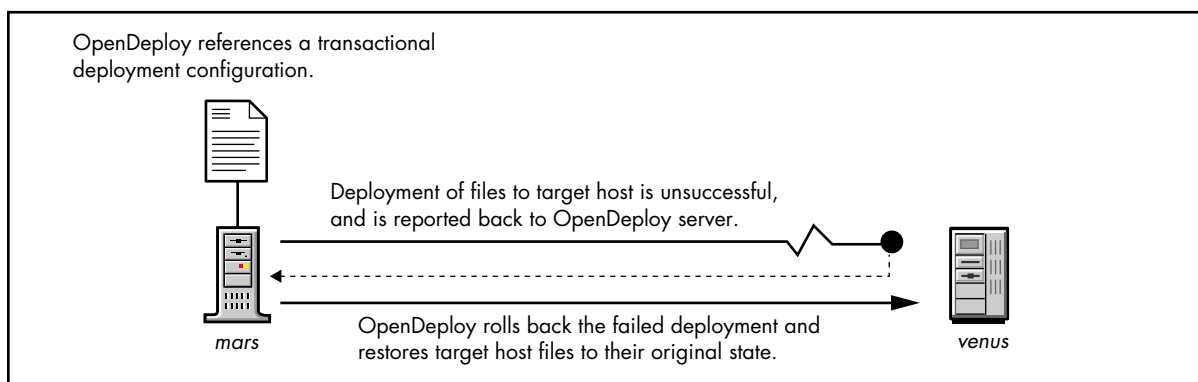


Figure 9: Transactional Deployment

Transactional deployments require free disk space equal to about three times the size of the deployed content during the course of the deployment. Performance time is also slower than other deployment methods.

## Reverse Deployments

In some cases, the files on a production server may be changed before those on the development server, bypassing the usual development workflow. For example, production servers might generate the following types of data:

- Web server log files
- Data files created via a CGI application
- Assets uploaded through a Web server application

As these production server files are generated, it might be important to deploy them back to the development server. *Reverse deployments* meet this requirement by comparing and moving files from a specified target server back to the source server.

In Figure 10, the production server has generated a number of production-related files that need to be deployed back to the development server. A reverse deployment configuration assigns the *reverse source* role to the production server, and the *reverse target* role to the development server. The deployment then takes place like any other deployment.

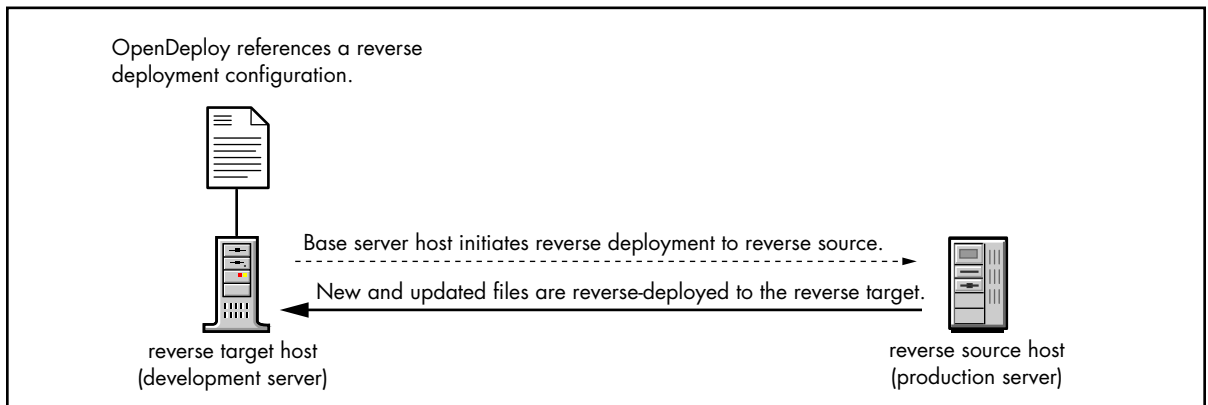


Figure 10: Reverse Deployment

## Access Rights and Privileges

In an organization with thousands of Web pages on hundreds of Web servers, deployment operations need to be carefully managed and restricted only to authorized users. OpenDeploy makes a distinction between those who need the authority to create and configure deployment configurations, and those who simply need the authority to start and monitor a specific deployment. To meet this need, there are *Administrator* and *User* roles that restrict what tasks an individual can perform using the OpenDeploy user interface. These roles apply both to what tasks can be performed involving any host in the OpenDeploy environment (the *system role*), and on specific hosts (the *server role*). Each individual OpenDeploy user can be assigned one or both of these roles. The individual must select which role they will be using when they login to the OpenDeploy user interface.

### Administrator Role

The Administrator role allows full access to OpenDeploy configuration and functionality. The Administrator role is authorized to perform any deployment operations including the following:

- Designate which users can invoke particular deployments.
- View, edit, and upload any deployment configuration.
- Schedule, start, cancel any deployment.
- Edit existing schedules.
- Monitor status of all deployments.
- Assign Administration and User roles to the appropriate individuals.

## User Role

The deployment User role authorizes an individual with User access to perform deployment operations on specific deployments (previously created by an individual in the Administrator role). Specifically, the User role can perform the following tasks for their deployments:

- Start and cancel deployments for which the individual is authorized.
- View and edit the XML-based configuration for which the individual is authorized.
- Create and edit schedules for which the individual is authorized.
- Monitor the status of all deployments.
- View the schedules for all deployments.



## Chapter 2

# Installation and Configuration

---

This chapter leads you through the tasks required to install the software components and to modify the configuration files to run OpenDeploy.

Refer to the *OpenDeploy Release Notes* for the latest information on the following installation-related topics:

- Supported operating systems
- Localized operating system support
- RAM requirements
- Storage requirements
- Patch requirements
- Supported browsers
- Compatibility between OpenDeploy releases
- Compatibility between OpenDeploy and other Interwoven products

## OpenDeploy Software Components

OpenDeploy has several software components, each of which must be installed on the appropriate server. Here are the components:

- Operations server software — this is software for managing the authentication of administrators and users.
- Administration server software — this is software for managing and generating the user interface.
- Base server software — this is the software which controls the management of deployments on the source host. This software permits the host server to send and received deployed files.
- Receiver software — this is the target host software that must be installed on each server designated only for receiving deployed files. Servers with the base server software installed do not need the receiver software.

The following section contains tips and strategies for installing each software component.

### Operations Server Software

The operations server software is installed on the operations server home directory, which is indicated in this manual by the term:

*ops-home*

The operations server software must be co-located on the same host with the administration server software.

### Windows

The default location of the operations server software on Windows is:

`C:\Interwoven\OpServer`

but you can select another location if you want.

## UNIX

On UNIX servers, you are prompted during the installation to designate a parent directory for the operations server software. Within its parent directory, the following directory structure is created for the operations server software:

*parent\_directory/OpenAPI/iwopenapi*

### Use with TeamSite OpenAPI

The operations server is a subset of the TeamSite OpenAPI. If you are using OpenDeploy in conjunction with TeamSite 5.0.1 or later, and already have OpenAPI installed, you must use the TeamSite OpenAPI rather than installing the operations server software that comes with OpenDeploy.

## Administration Server Software

The administration server software is installed on the administration server home directory, which is indicated in this manual by the term:

*admin-home*

The administration server software must be co-located on the same host with the operations server software.

## Windows

The default location for the administration server software on Windows is:

`C:\Interwoven\AdminServer`

but you can select another location if you want.

## UNIX

On UNIX servers, you are prompted during the administration server installation to designate a parent directory for the administration server software. Within its parent directory, the following directory structure is created for the administration server software:

*parent\_directory/AdminServer*

## Tomcat Server

One component of the administration server is the *Tomcat server*. The Tomcat server helps manage and generate the OpenDeploy user interface. When you install the administration server software, the installation program will ask you whether the Tomcat server is already installed on your host. If it is not already installed, you will be asked in what location you want to install it. Indicate where on the host the Tomcat server software should be installed.

You can also elect to use a different Tomcat server than the one installed as part of the OpenDeploy administration server software. However, this configuration is not officially supported. See “Using Different Administration Server Software” on page 92 for more information.

## opendeploy.war File

Another component of the administration server is the `opendeploy.war` file. This file generates various directories and files within the *admin-home* directory to support the user interface. This file is automatically installed as part of the administration server software component. However, in some cases it might be necessary to replace this file with a new or existing version. See “Refreshing the opendeploy.war File” on page 95 for more information.

## Base and Receiver Software

OpenDeploy base server and receiver software is installed in the OpenDeploy home directory, which is indicated in this manual by the term:

*od-home*

Installation of the base server and receiver software is essentially the same. Here is the criteria for where these software components should be installed.

- Base server software — install on the server designated as an OpenDeploy source host, a server capable of sending deployments to target hosts. The base server is also capable of receiving deployments from other source hosts.
- Receiver software — install on each server designated as an OpenDeploy target host, a server capable only of receiving deployments from the source host.

Because a server with the base server software installed can receive deployed files as well as send them, there is never a need to install both software components on the same server.

## Windows

The default location for the base server and receiver software on Windows is:

```
C:\Interwoven\OpenDeployNG
```

but you can select another location if you want.

## UNIX

On UNIX servers, you are prompted during the base server and receiver installation to designate a parent directory for the administration server software. Within its parent directory, the following directory structure is created for the base server or receiver software:

```
parent_directory/OpenDeployNG
```

# Software Installation Strategies

You can install all the OpenDeploy software components on a single server, known as a *single-host installation*, or you can spread them out among several servers, known as a *multi-host installation*. Your decision will be based on factors such as server availability and capability, and administrator availability and expertise.

## Single-Host Installation

Depending on the number and capabilities of your available servers, you might want to install two or more components, such as the source host, administration server, or operations server software on a single server.

To install all the required OpenDeploy software on a single server, you must perform separate installations for the following software components in this order:

- Operations software
- Administration server software
- Base server software

The OpenDeploy installation program will only install one software component at a time. Install the components and proceed to the next component when prompted.

Figure 11 shows the OpenDeploy environment based on a single-host installation, with the source host, administration, and operations software all residing on a single server.

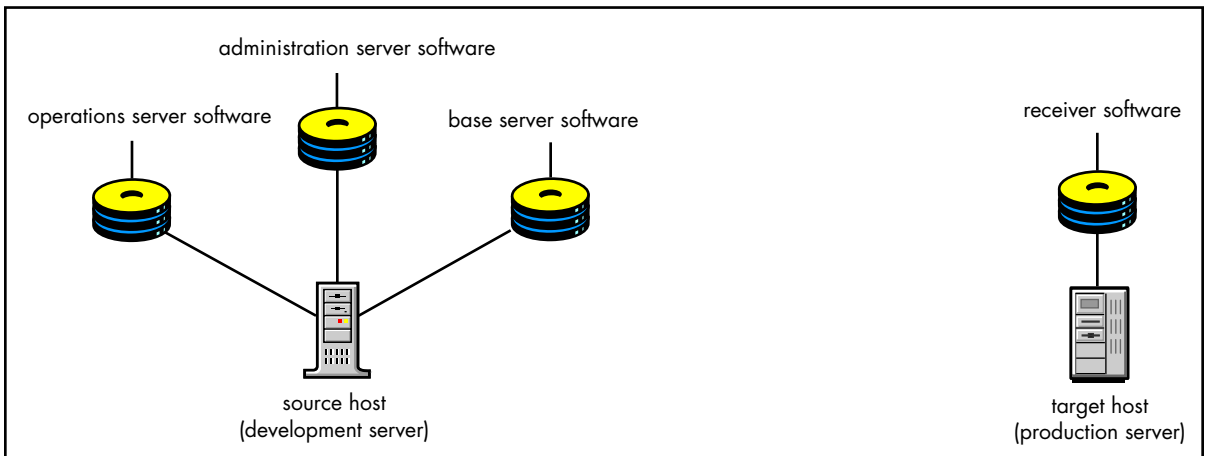


Figure 11: Single-Host Installation of OpenDeploy Software Components

## Multi-Host Installation

In other cases, it might be preferred to have the software components distributed among several different servers in a multi-host installation. This method is preferred if you do not have a single server capable of running multiple components. The administration and operations server software components must both reside on the same server machine. However, you can locate any number of base server and receiver software components on other server machines within the enterprise.

Figure 12 shows an example of a multi-host installation, where OpenDeploy software components are installed and running on separate servers.

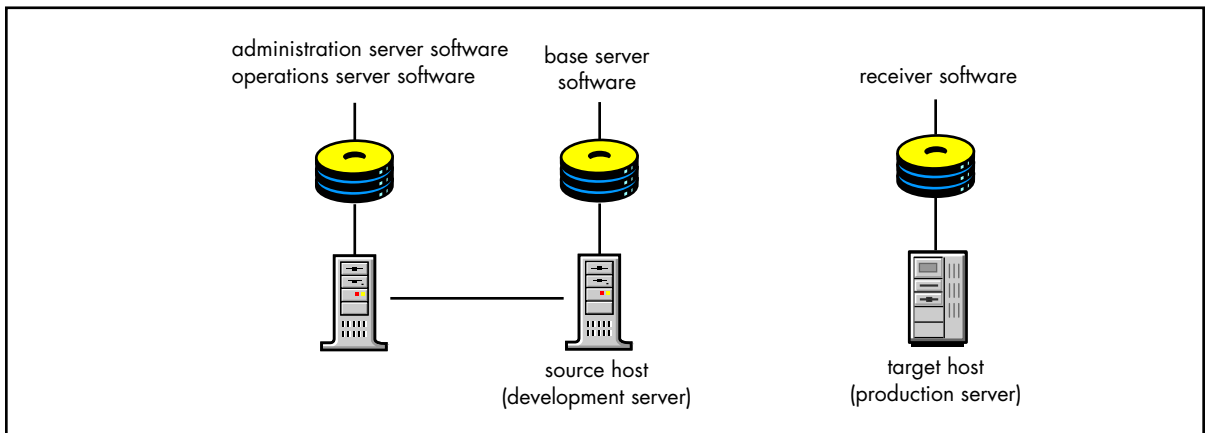


Figure 12: Multi-Host Installation of OpenDeploy Software Components

## Receive-Only Target Hosts

Servers that only need to receive deployed files, such as production servers, only require the receiver software to be installed.

## **RMI Registry Service Considerations**

One of the underlying services of Interwoven products such as OpenDeploy and TeamSite is the RMI registry. Interwoven products using this service can work together without a problem. However, other software products that make use of the RMI registry can cause conflicts that might prevent OpenDeploy and other Interwoven products from working properly. It is strongly recommended that no other programs accessing the same RMI registry as the ones used by OpenDeploy and TeamSite be included in your OpenDeploy environment.

The RMI registry ports being used are:

- Operations server — 1099
- OpenDeploy — 9173 (default) or a port you specify during installation.

## **Installation**

Before you install OpenDeploy software on your servers, decide how you want to distribute the software components over your enterprise. See “Software Installation Strategies” on page 49 for information.

Install the appropriate components, one at a time, on the servers making up your OpenDeploy environment. Review “OpenDeploy Software Components” on page 46 to determine the requirements for each component. At the end of each successful software installation, the OpenDeploy installation program returns you to the main menu, where you can install another OpenDeploy software component.

## **Information Requirements**

The following sections describe the information and decisions you must provide for each software component installation.

## Operations Server

When installing the operations server component, you must specify where the operations server software is to be installed. You can accept the default installation location or specify your own. If you are using OpenDeploy in conjunction with TeamSite, you must use the TeamSite OpenAPI instead of the OpenDeploy operations server software. See “Using TeamSite OpenAPI with OpenDeploy” on page 57 and page 62 for Windows and UNIX, respectively, for more information.

## Administration Server

Installation of the administration server software includes the following tasks:

- (UNIX only) Indicating whether you want to install the administration server software for OpenDeploy or OpenSyndicate. Select the option for OpenDeploy.
- Specifying where the administration server component is to be installed. You can accept the default installation location or specify your own.
- Specifying the port number used to broadcast the user interface. The default port is 8081. The port number you choose must be included in the URL that you enter to access the OpenDeploy user interface. See “OpenDeploy User Interface” on page 119 for more information.

## Base Server

Installation of the base server software includes the following tasks:

- Accepting the license agreement.
- Specifying where the base server component is to be installed. You can accept the default installation location or specify your own.
- Specifying the port number for RMI registry service. The default port number is 9173. To avoid conflicts with the RMI registry service, it is strongly recommended that you do not include non-Interwoven applications that might need to use these ports in your OpenDeploy environment.
- Specifying the port number for incoming deployments. The default port number is 20014.
- (Windows only) Specifying the location for the TMP system environmental variable.

- Indicating whether the server host where the operations server software resides is a Windows or UNIX server. If you are using OpenDeploy in conjunction with TeamSite, you must use the OpenAPI instead of the operations server whether the base server software is installed on the same host as TeamSite, or on a different host within the OpenDeploy environment.
- Indicating the user name and domain (Windows only) assigned to the bootstrap administrator. The domain is the domain for the operations server. The user is the user defined on the operations server host and that has the `od-admin` role. This user and domain will be added as the bootstrap administrator to the service configuration file (`deploy.cfg`) of the OpenDeploy host.
- Indicating whether or not you want to use the default scheduler database. See “Defining the Scheduler Database” on page 78 for more information on this feature.

If you accept the default scheduler database, you will be prompted to accept its default name or enter another. The default name is `schedDB`.

If you do not accept the default scheduler database because you are using your own, you will be prompted for the following information:

- Name of the database (DBName). The default name is `schedDB`.
  - Name of the database user (DBUSR). The default name is `sa`.
  - Password (PWD). Enter a password to use when accessing the database.
- Perform bootstrap administration configuration. See “Configuring the Bootstrap Administrator” on page 67 for more information.

## Receiver

Installation of the receiver software includes the following tasks:

- Accepting the license agreement.
- Specifying where the receiver component is to be installed. You can accept the default installation location or specify your own.
- Specifying the port number for RMI registry service. The default port number is 9173. To avoid conflicts with the RMI registry service, it is strongly recommended that you do not include non-Interwoven applications that might need to use these ports in your OpenDeploy environment.

- Specifying the port number for incoming deployments. The default port number is 20014.
- (Windows only) Specifying the location for the TMP system environmental variable.
- Indicating whether the server host where the operations server software resides is a Windows or UNIX server. If you are using OpenDeploy in conjunction with TeamSite, you must use the OpenAPI instead of the operations server.
- Indicating the user name and domain (Windows only) assigned to the bootstrap administrator. The domain is the domain for the operations server. The user is the user defined on the operations server host and that has the `od-admin` role. This user and domain will be added as the bootstrap administrator to the service configuration file (`deploy.cfg`) of the OpenDeploy host.
- Perform bootstrap administration configuration. See “Configuring the Bootstrap Administrator” on page 67 for more information.

### **Bootstrap Administrator**

Configure your bootstrap administrator after you have completed installing your software components. You must perform this task before restarting your server. See “Configuring the Bootstrap Administrator” on page 67 for more information.

## **Installing OpenDeploy into a TeamSite Environment**

If you are installing OpenDeploy into an environment where TeamSite 5.5 or later is already installed, you must use the TeamSite OpenAPI rather than installing the operations server software that comes with OpenDeploy. See “Using TeamSite OpenAPI with OpenDeploy” on page 57 and page 62 for Windows and UNIX, respectively, for more information.

## Installing OpenDeploy Software on Windows

You must have administrator privileges to install OpenDeploy on your Windows server.

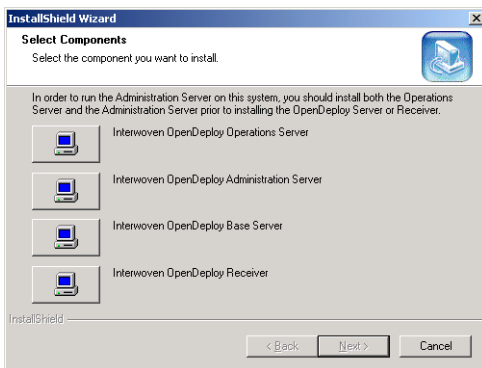
Close all other open applications before installing OpenDeploy software on your Windows server.

To install OpenDeploy on a supported Window NT or Windows 2000 server, follow these steps:

1. Insert the OpenDeploy CD-ROM and navigate to the CD-ROM's contents.
2. Double-click the installation file:

`ODWRAPPER551Buildxxxx.exe`

where `xxxx` indicates the build number of the final shipping product to start the installation program (Figure 13).



*Figure 13: Windows Installation Wizard*

The installation program presents you with the following options for installing the OpenDeploy software in order:

- Operations server
- Administration server
- Base server or receiver (you cannot install both on the same host)

3. Install the appropriate components, one at a time, on the servers making up your OpenDeploy environment. Review “OpenDeploy Software Components” on page 46 to determine the requirements for each component.

If you are installing OpenDeploy into an environment where TeamSite 5.0.1 or later is already present, you must use the TeamSite OpenAPI instead on the OpenDeploy operations server. Do not install the operations server software in this case. See “Using TeamSite OpenAPI with OpenDeploy” on page 57 for more information.

At the end of each successful software installation, the OpenDeploy installation program returns you to the main menu, where you can install another OpenDeploy software component. Be sure to wait until the installation program prompts you to install another component before proceeding.

### **Configuring the Bootstrap Administrator**

After you have completed installing your OpenDeploy software components, you need to configure a bootstrap administrator on the host where you installed your base server software. You must perform this task before restarting your server following installation of OpenDeploy software. After you have configured your bootstrap administrator, you can restart your server and log in to the OpenDeploy user interface. See “Configuring the Bootstrap Administrator” on page 67 for more information.

### **Using TeamSite OpenAPI with OpenDeploy**

If TeamSite 5.0.1 or later is present within your system environment, you must use the TeamSite OpenAPI instead of the OpenDeploy operations server.

To configure the TeamSite OpenAPI to work with OpenDeploy, follow these steps:

1. Ensure the OpenAPI server software is running. If OpenAPI is not running, you can start it by navigating to the following location:

```
iw-home\bin
```

where *iw-home* is the home directory of your TeamSite software, and entering the following command at the prompt:

```
iwreset -ui
```



2. Navigate to the following location if you have not done so already:

```
iw-home\bin
```

3. Add the OpenDeploy roles to OpenAPI by entering the following commands at the prompt:

```
iwuser -c -role od-admin
```

```
iwuser -c -role od-user
```

If the `od-admin` and `od-user` roles are already known to OpenAPI, the following message will appear:

```
Error-unable to add entry to database:Item already exists.
```

You can disregard this message and proceed to the next step.

4. Configure the bootstrap administrator for OpenDeploy. See “Configuring the Bootstrap Administrator” on page 67 for more information.
5. Restart the OpenAPI server software by entering the following command from *iw-home\bin*:

```
iwreset -ui
```

TeamSite must be running for OpenAPI to function properly. If you need to start TeamSite in order to start OpenAPI, enter the following command from *iw-home\bin*:

```
iwreset -a
```

### Using Different Administration Server Software

You can also elect to use a different Tomcat server than the one installed as part of the OpenDeploy administration server software. However, this configuration is not officially supported. See “Using Different Administration Server Software” on page 92 for more information.

## Setting the TMP System Environmental Variable on Windows

Following the installation of the OpenDeploy software on a Windows host, ensure that the TMP system environment variable is set. Normally this task is included as a prompt during installation. However, if your TMP environment settings are changed or lost, you can reset them using the following method.

To set the TMP system environment variable on a Windows host, follow these steps:

1. Open the System Properties window. This process may differ depending on which version of Windows you are using.
2. Open the Environment Variables window from the System Properties window. This process may differ depending on which version of Windows you are using.
3. Scroll through the System variables list.  
If TMP is present in the list, then the variable is set. You do not need to perform any further steps. Ensure that the TMP variable you see resides in the **System variables** list, not the **User variables** list.  
If TMP is not present, then continue with the rest of the steps to set the variable.
4. Add a new system variable called TMP and assign its value as the path to the location where you want to put temporary files. Usually this location is C:\Temp. This process may differ depending on which version of Windows you are using.
5. Click **OK** and exit the System Properties window.

If you have installed the base server software, as well as all other required software components, you must now configure your bootstrap administrator. Perform this task before restarting your host server. See “Configuring the Bootstrap Administrator” on page 67 for more information.

If you have installed the receiver software, you can restart your host server.

## Installing OpenDeploy Software on UNIX

Before you install OpenDeploy software on your UNIX servers, you must decide how you want to distribute the software components over your enterprise. See “Software Installation Strategies” on page 49 for information.

You must have root privileges to install OpenDeploy on your UNIX server.

To install OpenDeploy on a supported UNIX server, follow these steps:

1. Insert the OpenDeploy CD-ROM and navigate to the CD-ROM’s contents.
2. Copy the installation file:

```
IWOVopendeploy.5.5.1.Buildxxxx.tar.gz
```

where *xxxx* indicates the build number of the final shipping product, from the CD-ROM to its parent directory on your UNIX server.

The location where you copy the installation file will only be a working directory for the installation. You can install the software in any location on the server.

3. Unzip the installation file by entering the following command at the prompt:

```
gzip -d IWOVopendeploy.5.5.1.Buildxxxx.tar.gz
```

4. Untar the installation file by entering the following command at the prompt:

```
tar xvf IWOVopendeploy.5.5.1.Buildxxxx.tar
```

The following files will be created:

- IWOVadmin.tar
- IWOVopendeploy.5.5.1.Buildxxxx.tar
- IWOVopserver.tar
- ODNGr.tar
- ODNGs.tar
- Readme
- startinstall

OpenDeploy includes an installation script that provides menu selections for each component you want to install.

5. Start the installation script by entering the following command at the prompt:

**./startinstall**

The OpenDeploy installation script (Figure 14) appears:

```
#####
###
#Welcome to install Interwoven OpenDeploy product!                               #
#In order to run the Administration Server on this system, you should install
#
#both the Operations Server and the Administration Server prior to installing
#
#the OpenDeploy Server or Receiver.                                              #
#####
###
Please choose one of the following components.
Interwoven Operations Server ----->1
Interwoven OpenDeploy Administration Server ----->2
Interwoven OpenDeploy Base Server ----->3
Interwoven OpenDeploy Receiver ----->4
Quit the installation ----->5

Choose option:
```

Figure 14: UNIX Installation Script

Install each component on your server by entering the corresponding number. For best results, you should install each component in the following order:

- Operations server (unless TeamSite 5.0.1 or later is already installed)
- TeamSite OpenAPI (if TeamSite 5.0.1 or later is already installed)
- Administration server
- Base server or receiver

The installation script will prompt you for configuration information depending on which component you are installing. See “Information Requirements” on page 52 for more information. You will also be asked to provide an absolute path to a location under which the software component will be installed. For example, if you were installing the administration server software, and entered the absolute path:

```
/etc/OpenDeployNG
```

then the administration server software would be installed in the following location:

```
/etc/OpenDeployNG/AdminServer
```

You might want to keep all your OpenDeploy software components in the same home directory. If so, enter the same absolute path for each component you install. The home directories of each component (*ops-home*, *admin-home*, and *od-home*) will reside within this location.

When you have completed the installation, delete the installation file to save room on your server.

After you have installed all the base server or receiver software, as well as any other OpenDeploy components, you must configure your bootstrap administrator before restarting the host. See “Configuring the Bootstrap Administrator” on page 67 for more information.

### Using TeamSite OpenAPI with OpenDeploy

If TeamSite 5.0.1 or later is present within your system environment, you must use the TeamSite OpenAPI instead of the OpenDeploy operations server.

To configure the TeamSite OpenAPI to work with OpenDeploy, follow these steps:

1. Ensure the OpenAPI server software is running. If OpenAPI is not running, you can start it by navigating to the following location:

```
iw-home/bin
```

where *iw-home* is the home directory of your TeamSite software, and entering the following command at the prompt:

```
./iwreset -ui
```

2. Navigate to the following location if you have not done so already:

```
iw-home/bin
```

3. Add the OpenDeploy roles to OpenAPI by entering the following commands at the prompt:

```
./iwuser -c -role od-admin
```

```
./iwuser -c -role od-user
```

If the `od-admin` and `od-user` roles are already known to OpenAPI, the following message will appear:

```
Error-unable to add entry to database:Item already exists.
```

You can disregard this message and proceed to the next step.

4. Continue installing your OpenDeploy software components. If you have completed the installation, configure your bootstrap administrator before restarting your server. See “Configuring the Bootstrap Administrator” on page 67 for more information.
5. Restart the OpenAPI server software by entering the following command from *iw-home*\bin:

```
./iwreset -ui
```

TeamSite must be running for OpenAPI to function properly. If you need to start TeamSite in order to start OpenAPI, enter the following command from *iw-home*/bin:

```
./iwreset -a
```

### Using Different Administration Server Software

You can also elect to use a different Tomcat server than the one installed as part of the OpenDeploy administration server software. However, this configuration is not officially supported. See “Using Different Administration Server Software” on page 92 for more information.

## Upgrading From a Previous Release

If you are upgrading from OpenDeploy 5.0.1 or later to the current release, you can keep many of your existing settings, such as those for the ports, bootstrap administrator information, and scheduler database, by installing the upgrade in the same location as the existing software. Important configuration files, such as deployment configurations are also preserved for use with the upgraded software. Upgrading is the same for both Windows and UNIX hosts. It is not possible to upgrade from OpenDeploy releases earlier than 5.0.1. However, OpenDeploy 5.5.1 can co-reside on the same host as OpenDeploy 4.x.

For each OpenDeploy software component you want to upgrade, the installer will check for existing software already installed. If it finds existing software, it will prompt you to allow it to remove it as a preparation for installing the new software. Allow the installer to continue.

Upon completion of the de-installation, OpenDeploy will then prompt you to start installation of the upgrade software. Begin the installation. You will be prompted for whatever information OpenDeploy requires to complete the upgrade.

You must install the new software into the same location as the existing software to retain your original settings. Installing a software component into a new location will cause the OpenDeploy installer to treat it as a new installation, and you will be required to re-enter all the configuration settings. Those existing configuration files that are preserved will remain in their original locations. You must copy them manually to your new software location.

## Modifying the Service Configuration File

The *service configuration file* for the base server and receiver components contains service-related configuration information. Normally, you do not need to modify this file. However, if you change the default names of the base server, receiver, or nodes configuration files, then you must edit the associated service configuration file to reflect these updated names. Unlike other configuration files, you cannot change the name of this from its default name.

This service configuration file resides in the following location:

*od-home/etc/deploy.cfg*

The contents of service configuration file must be ASCII text, including any modifications you make. No multi-byte or non-ASCII characters are allowed.

## Referencing the Base Server and Receiver Configuration Files

The `Deploy.serverConfig` attribute's value must point to the appropriate server host configuration file depending on whether the server has base server or the receiver software installed. If you are using the default configuration files for these types of hosts, then the attribute value would be one of the following:

base server — `Deploy.serverConfig: odbase.xml` *or*

receiver — `Deploy.serverConfig: odrcvr.xml`

If the server host file has a different name than the default name listed here, then the `Deploy.serverConfig` attribute's value must point to that file. You also must ensure that the server host file being pointed to resides in the *od-home/etc* directory on the host.

## Referencing the Nodes Configuration Files

The `Deploy.serverNodesConfig` attribute's value must point to the nodes configuration file for the server host. If you are pointing to the default nodes configuration file, the attribute would be:

```
Deploy.serverNodesConfig: odnodes.xml
```

If the nodes configuration file has a different name than the default files listed here, then the `Deploy.serverNodesConfig` attribute's value must point to that file. You also must ensure that the nodes configuration file being pointed to resides in the `od-home/etc` directory on the host.

## Referencing the Bootstrap User Name

The `Deploy.bootstrapUserName` attribute allows you to define your own bootstrap administrator. This is an optional feature in the service configuration file. See “Configuring the Bootstrap Administrator” on page 67 for more information. The format is:

Windows — `Deploy.bootstrapUserName: domain-name\username` or

UNIX — `Deploy.bootstrapUserName: username`

## Configuring the Bootstrap Administrator

When you first log into OpenDeploy, you must do so as the *bootstrap administrator*. As the bootstrap administrator, you can start performing role access management by assigning OpenDeploy Administrator and User roles to the appropriate individuals who require access to the OpenDeploy server. The bootstrap administrator name must be a valid operating system user account on the operations server host.

By default, both the operations server and the base server automatically create the following bootstrap administrators in their `od-admin` role:

Windows — `hostname\Administrator` *or*

UNIX — `root`

If you are installing both the operations server and base server software on the same Windows host, the bootstrap administrator will be the same in both servers. Otherwise, the hostnames of the respective hosts will not match and you will need to define your own bootstrap administrator. You always have the option of configuring the bootstrap administrator to any user account you want.

Adding the bootstrap administrator to the operations server varies depending on whether you are using the operations server that comes with OpenDeploy, or you are using the OpenAPI that comes with TeamSite. If you are using OpenDeploy in conjunction with TeamSite, you must use the TeamSite OpenAPI instead of the OpenDeploy operations server. Configuring the bootstrap administrator is the same for both Windows and UNIX, and requires the following separate tasks:

- Stopping any OpenDeploy services or daemons currently running.
- Adding the bootstrap administrator account information to the OpenDeploy operations server. Perform this task only if TeamSite 5.0.1 or later is not present in your environment.
- Adding the bootstrap administration account information to the base server. Perform this task only if you are using OpenDeploy in conjunction with TeamSite OpenAPI, which is used with TeamSite 5.0.1 and later.
- Restarting the OpenDeploy services or daemons.

## Stopping the OpenDeploy Services and Daemons

Stop any of the OpenDeploy services or daemons that are running before beginning the bootstrap administrator configuration. See “Stopping OpenDeploy” on page 113 for more information.

## Adding Bootstrap Administrator to the Operations Server

Perform this task only if you are using OpenDeploy outside of a TeamSite 5.0.1 or later environment. If you are using OpenDeploy in conjunction with TeamSite 5.0.1 or later, you must add the bootstrap administrator to the TeamSite OpenAPI. This is covered in the following section.

To add the bootstrap administrator to the operations server, follow these steps:

1. Open the `od-admin.uid` file using your favorite text editor. This file resides in the following location:

*ops-home/iwopenapi/roles/od-admin.uid*

2. Add a line to the end of the file containing your boot strap administrator name and domain (Windows only). For example:

Windows — `WORKGROUP\jdoe` *or*

UNIX — `jdoe`

3. Save and close the file.

## Adding Bootstrap Administrator to TeamSite OpenAPI

Perform this task only if you are using OpenDeploy in conjunction with TeamSite 5.0.1 or later. If TeamSite 5.0.1 or later is not present in your environment, you must add the bootstrap administrator to the OpenDeploy operations server. This is covered in the previous section.

To add the bootstrap administrator to the TeamSite OpenAPI server, follow these steps:

1. Open the `od-admin.uid` file using your favorite text editor. This file resides in the following location:

*iw-home/conf/roles*

where *iw-home* is the home directory of TeamSite.

2. Add a line to the end of the file containing your bootstrap administrator name. For example:

Windows — WORKGROUP\jdoe *or*

UNIX — jdoe

3. Save and close the file.

### **Adding Bootstrap Administrator to the OpenDeploy Base Server**

To add the bootstrap administrator to OpenDeploy base server, follow these steps:

1. Open the `deploy.cfg` file using your favorite text editor. This file resides in the following location:

`od-home/etc`

2. Review the `Deploy.bootstrapUserName` attribute to ensure it has the bootstrap administrator name and domain (Windows only), and make any necessary modifications. For example:

Windows — WORKGROUP\\jdoe *or*

UNIX — jdoe

For Windows, two backslashes (“\\”) are required to separate the domain and the user name.

During the initial installation of the base server or receiver software, you are prompted to provide a bootstrap name and domain. For upgrades, OpenDeploy uses the existing bootstrap information. You should check to see if the current values are still what you want.

### **Start the OpenDeploy Services and Daemons**

Start your OpenDeploy services or daemons in the following order:

- Operations or TeamSite OpenAPI server. TeamSite must already be running to start OpenAPI.
- Administration server
- OpenDeploy base server

See “Starting OpenDeploy” on page 109 for more information.

## Defining Target Host Nodes

When the base server or receiver component is installed on your OpenDeploy source host, it includes the *nodes configuration file*. The nodes configuration file contains listings and information for each host within the OpenDeploy environment capable of receiving files from the source host, including:

- Logical host name
- Physical host name
- Listener port

When the base server or receiver software is installed, the nodes configuration file is created with a single node listing for the source host itself. This allows you to configure the source host to deploy files to itself. However, you must add all other target nodes into the nodes configuration file manually. Any target node you add must have the base server or receiver software installed on it.

The name and location of the default nodes configuration file that is installed with the OpenDeploy base server software is:

```
od-home/etc/odnodes.xml
```

This file can be renamed. However, that name must be referenced in the OpenDeploy service configuration file (`deploy.cfg`) as the value of the `Deploy.serverNodesConfig` attribute. See “Modifying the Service Configuration File” on page 65 for more information.

The nodes configuration file is an XML-based file derived from the nodes configuration DTD. You can find this DTD on your base server host at the following location:

```
od-home/etc/dtd/odnodes.dtd
```

Any modifications to the nodes configuration file you make must conform to the nodes configuration DTD.

## Encoding

The encoding for the nodes configuration file can be encoding other than UTF-8. For example, if a value in the file contains Japanese characters, the encoding will need to be:

```
<? xml version="1.0" encoding="SHIFT_JIS" ?>
```

For French and German, the encoding value would be:

```
<? xml version="1.0" encoding="ISO-8859-1" ?>
```

Check what the appropriate value is for any non-ASCII characters and modify the nodes configuration file encoding as needed. If no encoding is specified, UTF-8 will be used by default.

## Logical vs. Fully Qualified Host Names

OpenDeploy allows you to map a simple logical name to the fully qualified DNS host name or IP address within the nodes configuration file residing on the source host. You can use any logical name you want, as long as it is unique.

Using a brief logical name relieves you from having to enter lengthy host names into each deployment configuration. Additionally, in many organizations the deployment group is separate and distinct from the network group. Administratively, if there are changes in the network, such as a server being replaced, it will not affect the deployment configuration, just the nodes configuration file where the mapping of names occurs.

## Specifying Host Nodes

The nodes configuration file contains the `nodeSet` element, which houses the individual nodes listings. You can provide an ID value for the name attribute of the `nodeSet` element, for example:

```
name="od_receiver_nodes"
```

or keep the default implied name.



Within this element you can add a separate instance of the `node` element for each host within the OpenDeploy environment. The `node` element has the following attributes:

- `name` — the logical name of the host as it appears in OpenDeploy configuration files. For example:

```
name="venus"
```

- `host` — the fully qualified DNS host name or the IP address of the server. For example:

```
host="venus.mycompany.com" or
```

```
host="114.342.23.21"
```

- `port` — the listening port number used by the OpenDeploy server to send and received deployed files. For example:

```
port="20014"
```

The port number should match the value for the `bindPort` attribute of the `listenerProperties` element found in the base server configuration file. This value is typically the RMI port number you entered during installation of the base server software.

In the following example, if you had the following nodes in the OpenDeploy environment:

- *venus.mycompany.com* (IP address: 114.342.23.21)
- *jupiter.mycompany.com* (IP address: 114.342.23.22)
- *saturn.mycompany.com* (IP address: 114.342.23.23)

and you wanted to create and use logical names for them, you could map their host names to their fully qualified DNS host names this way:

```
<nodeSet name="od_receiver_nodes">
  <node name="venus"    host="venus.mycompany.com"    port="20014" />
  <node name="jupiter"  host="jupiter.mycompany.com"  port="20014" />
  <node name="saturn"   host="saturn.mycompany.com"   port="20014" />
</nodeSet>
```

You can also map logical names to their IP addresses in the following way:

```
<nodeSet name="od_receiver_nodes">
  <node name="venus"    host="114.342.23.21"    port="20014" />
  <node name="jupiter"  host="114.342.23.22"    port="20014" />
  <node name="saturn"   host="114.342.23.23"    port="20014" />
</nodeSet>
```

You can even mix DNS host names and IP addresses:

```
<nodeSet name="od_receiver_nodes">
  <node name="venus"    host="venus.mycompany.com"    port="20014" />
  <node name="jupiter"  host="jupiter.mycompany.com"  port="20014" />
  <node name="saturn"   host="114.342.23.23"          port="20014" />
</nodeSet>
```

See “Logical vs. Fully Qualified Host Names” on page 71 for more information on the use of logical names.

### Case Insensitivity of Host Logical Names Allowed

The logical names of hosts listed in the nodes configuration file and hosts listed as the `nodeRef` element’s `useNode` attribute value in a deployment configuration is case-insensitive. For example, if you had the following host entry in the nodes configuration file:

```
<node name="venus" host="venus.mycompany.com" port="20014" />
```

and you referenced this host in a deployment configuration as:

```
<nodeRef useNode="Venus" />
```

the target host could still be located and the deployment could take place.

## Modifying the Base Server Configuration File

When the base server software is installed on your OpenDeploy source host, it includes the *base server configuration file*. For example:

```
<?xml version="1.0" encoding="UTF-8"?>

<deployServerConfiguration>

    <localNode host="mars.mycompany.com"/>

    <listenerProperties name="InterwovenOpenDeploy"
                      bindPort="20014"/>

    <transportProperties name="od"
                       bufferSize="8000"
                       />

    <teamsiteProperties version="5.0"/>

    <schedulerProperties jdbcDriverClass="org.hsql.jdbcDriver"
                        dbUrl="jdbc:HyperpersoniSQL:C:\Interwoven\OpenDeployNG\db\schedDB"
                        dbUser="sa"
                        dbPassword=""
                        />

    <allowedHosts>
        <node host="mars.mycompany.com">
            <allowedDirectories>
                <path name="C:\Interwoven\OpenDeployNG\tmp"/>
                <path name="C:\temp"/>
            </allowedDirectories>
        </node>
    </allowedHosts>

    <logRules maxBytes="32Mb"
             directory="C:\Interwoven\OpenDeployNG\log"
             level      ="verbose"
             />

</deployServerConfiguration>
```

This file contains the elements and attributes that provide base server configuration settings such as the following:

- Encoding
- Communications with other OpenDeploy software components and nodes
- Specifying the ability to use downward revision TeamSite software
- Scheduling information
- Allowed access for other source hosts to this OpenDeploy host
- Logging defaults
- Encryption settings
- Deployment job queueing
- Deploying to downward revision OpenDeploy target hosts

The base server configuration file contains some elements and attributes whose values serve as default settings in case the particular deployment configuration does not specify a needed value. You can modify the file as needed to meet your specific deployment needs.

The name and location of the default base server configuration file that is installed with the OpenDeploy base server software is:

*od-home/etc/odbase.xml*

This file can be renamed. However, that name must be referenced in the OpenDeploy service configuration file (*deploy.cfg*) as the value of the *Deploy.serverNodesConfig* attribute. See “Modifying the Service Configuration File” on page 65 for more information.

The base server configuration file is an XML-based file derived from the deployment server DTD. You can find this DTD on your base server host at the following location:

*od-home/etc/dtd/odserver.dtd*

Any modifications to the base server configuration file you make must conform to the deployment server configuration DTD.

Typically, after the base server configuration file is configured, you do not have to modify it again unless there are changes to your host server itself, or to the network over which it deploys files. For a complete list of all elements and attributes that are required, or options within the base server configuration file, refer to the *OpenDeploy Reference*. For a basic discussion of XML structure and syntax, see “Understanding the Configuration DTDs” on page 195.

## Encoding

The encoding for the nodes configuration file can be encoding other than UTF-8. For example, if a value in the file contains Japanese characters, the encoding will need to be:

```
<? xml version="1.0" encoding="SHIFT_JIS" ?>
```

For French and German, the encoding value would be:

```
<? xml version="1.0" encoding="ISO-8859-1" ?>
```

Check what the appropriate value is for any non-ASCII characters and modify the nodes configuration file encoding as needed. If no encoding is specified, UTF-8 will be used by default.

## Communicating with Source and Target Hosts

Communication between OpenDeploy hosts is required for comparing files and for sending and receiving deployed files. To assist in establishing and optimizing this communication, you can set a variety of communication-related settings in the base server configuration file.

### Specifying the Communication Port

The OpenDeploy host uses the bind port to communicate with other hosts when comparing files, and sending and receiving deployed files. The bind port number is listed in the base server configuration file, where it is specified as the `listenerProperties` element's `bindPort` attribute. For example:

```
<deployServerConfiguration>
  ...
  <listenerProperties
    name="InterwovenOpenDeploy"
    bindPort="20014" />
  ...
</deployServerConfiguration>
```

The name attribute value is fixed as `InterwovenOpenDeploy`. The default bind port is 20014, but you can set it to any available port. However, if the bind port value is changed, any other source host deploying to your OpenDeploy host must update their nodes configuration file with this new port value.

For ease of port management, you might want to use the same bind port number for all of your OpenDeploy hosts. The bind port value is also used in other configurations, such as the nodes configuration file.

### Setting the File Transport Buffer Size

You can set a default buffer size in bytes for transporting files to and from your OpenDeploy server. This amount is specified in the `bufferSize` attribute of the `transportProperties` element. For example:

```
<deployServerConfiguration>
  ...
  <transportProperties bufferSize="8000" />
  ...
</deployServerConfiguration>
```

### Specifying Downward Revision TeamSite Releases

If your OpenDeploy base server and TeamSite software are of the same release, no additional OpenDeploy configuration is required to allow them to work with each other. However, if you are using a supported downward revision release of TeamSite, you must modify the base server configuration to enable OpenDeploy to use that TeamSite version for TeamSite comparison deployments.

You must specify the TeamSite version as a value for the `teamsiteProperties` element's `version` attribute. You only need to include the first two numbers of the TeamSite release, for example:

```
<deployServerConfiguration>
  ...
  <teamsiteProperties version="5.0" />
  ...
</deployServerConfiguration>
```

Inclusion of the version attribute is only necessary if a supported downward revision of TeamSite is in your OpenDeploy environment. Otherwise, you can omit this attribute from your deployment configuration. Refer to your *OpenDeploy Release Notes* for a listing of supported TeamSite releases.

## Defining the Scheduler Database

The OpenDeploy base server software requires a JDBC-compliant database to allow scheduling information to be stored in cases of server shutdown or similar situations. During installation, you are prompted whether you want to use the default database included with the OpenDeploy software, or use a different one of your own. The database packaged with OpenDeploy is Hypersonic SQL. You can reconfigure the OpenDeploy base server configuration to access a different database of your own choosing through the `schedulerProperties` element.

```
<deployServerConfiguration>
  ...
  <schedulerProperties ... />
  ...
</deployServerConfiguration>
```

The `schedulerProperties` contains the following attributes:

- `jdbcDriverClass` — enter the JDBC Java class that is used to communicate to the RDBMS. The default value is `org.hsqldb.jdbcDriver`, the Hypersonic SQL database.
- `dbUrl` — enter the Web URL to the scheduler database. See “URL Choices” on page 79 for more information.
- `dbUser` — enter the user account name for access to the scheduler database.
- `dbPassword` — enter the password to the scheduler database.
- `isClearPassword` — indicate whether or not the value of the `dbPassword` attribute is contained as unencoded plain text in the deployment configuration file. By default, it is assumed that the `dbPassword` value is an encoded string. However, if the `isClearPassword` attribute value is `yes`, then this password will be in plain text. The default implied value is `no`. The Hypersonic SQL database that is installed with the base server software does not require a password.

The default configuration for the Hypersonic SQL database is the following:

```
<schedulerProperties
  jdbcDriverClass="org.hsql.jdbcDriver"
  dbUrl="jdbc:HypersonicSQL:od-home/db/schedDB"
  dbUser="sa"
  dbPassword=""
/>
```

### URL Choices

You have two choices for values in the `dbUrl` attribute when you are using the default Hypersonic SQL database:

- In-Memory
- Stand-Alone (required to allow your scheduled entries to persist across restarts)

If you are using another database, these features will not be available.

### In-Memory

The URL for an In-Memory database is:

```
jdbc:HypersonicSQL:.
```

If you use In-Memory, the data will be lost if the server quits.

### Stand-Alone

In a Stand-Alone URL, the server stores the data in flat files. If the server restarts, the data will not be lost. The URL for Stand-Alone database is:

```
jdbc:HypersonicSQL:db_name
```

where *db\_name* is the filename, including path, of the database you entered or accepted during installation.



The database you use will connect with the following files:

- *db\_name.properties*
- *db\_name.data*
- *db\_name.script*

created in the location where the server was started.

The default database that comes with OpenDeploy has the following value:

```
dbURL="jdbc:HypersonicSQL:od-home/db/schedDB"
```

and connects with the following files:

- *schedDB.properties*
- *schedDB.data*
- *schedDB.script*

If you entered your own database *foo*, the *dbURL* attribute would be:

```
dbURL="jdbc:HypersonicSQL:path/foo"
```

and would connect with the following files:

- *foo.properties*
- *foo.data*
- *foo.script*

created in the home directory for *foo*.

### Using Third-Party JDBC Drivers with the OpenDeploy Scheduler

To use third-party JDBC drivers with the OpenDeploy scheduler, follow these steps:

1. Shut down the OpenDeploy base server by stopping its service or daemon.
2. Manually create the table entries in the database. The data definition language (DDL) scripts needed for creating the table are listed in the following section.
3. Provide driver-specific information in the `schedulerProperties` element of the base server configuration file. This element contains the following attributes:
  - `jdbcDriverClass` — specifies the JDBC Java class that is used to communicate to the RDBMS.
  - `dbUrl` — specifies the Web URL to the scheduler database.
  - `dbUser` — specifies the user account name for access to the scheduler database.
  - `dbPassword` — specifies the password to the scheduler database.
  - `isClearPassword` — indicates whether or not the value of the `dbPassword` attribute is contained as unencoded plain text in the deployment configuration file. By default, it is assumed that the `dbPassword` value is an encoded string. However, if the `isClearPassword` attribute value is `yes`, then this password will be in plain text. Default value is `no`.
4. Add the vendor-specific `.jar` file in the classpath.

Following are the DDL scripts for creating the database table entries:

```
CREATE TABLE IWOV_SCHEDULE (PK VARCHAR(128) NOT NULL, SCHEDULED_ITEM TIMESTAMP NOT NULL, TIME_EXPRESSION VARCHAR(128), END_TIME TIMESTAMP, REPEAT_COUNT INTEGER, TYPE VARCHAR(128), CALENDAR VARCHAR(128), STATE VARCHAR(128) NOT NULL, LISTENERS VARCHAR(128), DAYINV VARCHAR(128), DEPLOYNAME VARCHAR(128), DEPOYSUFFIX VARCHAR(128), DESCRIPTION VARCHAR(128), ENDTIME VARCHAR(128), HOURINV VARCHAR(128), KEYVALUES VARCHAR(128), LOGLEVEL VARCHAR(128), MINUTEINV VARCHAR(128), MONTHDAYS VARCHAR(128), NO_RECUR VARCHAR(128), OWNER VARCHAR(128), PAD1 VARCHAR(128), PAD2 VARCHAR(128), PAD3 VARCHAR(128), PAD4 VARCHAR(128), PAD5 VARCHAR(128), PAD6 VARCHAR(128), RECURSET VARCHAR(128), REPAIR VARCHAR(128), REPEAT VARCHAR(128), STARTDEPLOY BIT, STARTTIME VARCHAR(128), SYNCH VARCHAR(128), VERIFY VARCHAR(128), WEEKDAY VARCHAR(128), WEEKINV VARCHAR(128), YEARINV VARCHAR(128))
```

```
CREATE TABLE IWOV_SCHEDULE_PK (NEXT_PK VARCHAR(128) NOT NULL)
```

```
CREATE TABLE IWOV_QUEUE (PK VARCHAR(128) NOT NULL, NAME_FK VARCHAR(128) NOT NULL, PRIORITY INTEGER NOT NULL, STATE VARCHAR(128) NOT NULL, DAYINV VARCHAR(128), DEPLOYNAME VARCHAR(128), DEPOYSUFFIX VARCHAR(128), DESCRIPTION VARCHAR(128), ENDTIME VARCHAR(128), HOURINV VARCHAR(128), KEYVALUES VARCHAR(128), LOGLEVEL VARCHAR(128), MINUTEINV VARCHAR(128), MONTHDAYS VARCHAR(128), NO_RECUR VARCHAR(128), OWNER VARCHAR(128), PAD1 VARCHAR(128), PAD2 VARCHAR(128), PAD3 VARCHAR(128), PAD4 VARCHAR(128), PAD5 VARCHAR(128), PAD6 VARCHAR(128), RECURSET VARCHAR(128), REPAIR VARCHAR(128), REPEAT VARCHAR(128), STARTDEPLOY BIT, STARTTIME VARCHAR(128), SYNCH VARCHAR(128), VERIFY VARCHAR(128), WEEKDAY VARCHAR(128), WEEKINV VARCHAR(128), YEARINV VARCHAR(128))
```

```
CREATE TABLE IWOV_QUEUE_NAMES (PK VARCHAR(128) NOT NULL, NAME VARCHAR(128) NOT NULL, STATE VARCHAR(128) NOT NULL)
```

## Specifying Allowed Hosts for Received Deployments

You must specify which source hosts in the OpenDeploy environment can deploy files to your OpenDeploy server in the `allowedHosts` element of the base server configuration file.

```
<deployServerConfiguration>
  ...
  <allowedHosts>
    ...
  </allowedHosts>
  ...
</deployServerConfiguration>
```

When a deployment is started, the receiving host compares the host name of the deploying host (as specified in the `host` attribute value of the deployment's `localNode` element) with the host name values residing within the `allowedHosts` element. Matches can be made with any combination of case-insensitive fully qualified DNS host names and IP addresses. In the following example, only the hosts *jupiter.mycompany.com* and *114.342.23.23* can deploy files to this OpenDeploy server:

```
<allowedHosts>
  <node host="jupiter.mycompany.com" />
  <node host="114.342.23.23" />
</allowedHosts>
```

### Checking for Allowed Hosts

When a deployment is initiated, the receiving OpenDeploy host will match the name of the sending host with each of the nodes listed in the `allowedHosts` element of its base server or receiver configuration file. The host names are compared as case-insensitive strings. If this comparison results in a match, the deployment is allowed to continue.

If the string comparison does not result in a match, OpenDeploy will attempt to determine and match the IP addresses of the sending host with the hosts listed in the `allowedHost` element. If this comparison results in a match, the deployment is allowed to continue. If the comparison fails, the deployment is not allowed to take place.

There is no IP address checking during reverse deployments. Only the string comparisons will take place.

## Managing IP Address Checking

To ensure that IP address checking will occur, purposely use a mix of DNS names and IP addresses in your configurations. In some cases you might not want IP address checking to occur, such as if you have a DNS problem, or in the case of DHCP where IP changes dynamically. If you do not want IP address checking to occur, use the same host name strings in your source and target configurations.

## Specifying Allowed Directories for Deployments

You can match specific target directories on your OpenDeploy host with each source host listed in the `allowedHosts` element. This way you can control where deployed files from specific source hosts are placed on your target host.

You must add the `allowedDirectories` element and at least one occurrence of the `path` element for each specified node in the `allowedHosts` element. The path of the allowed directory is specified as the value for the `name` attribute of the `path` element.

In the following example:

```
<allowedHosts>
  <node host="jupiter.mycompany.com">
    <allowedDirectories>
      <path name="C:\reports\monthly" />
    </allowedDirectories>
  </node>

  <node host="114.342.23.23">
    <allowedDirectories>
      <path name="C:\reports\yearly" />
    </allowedDirectories>
  </node>
</allowedHosts>
```

the source host *jupiter.mycompany.com* is tasked with deploying monthly reports and the source host *114.342.23.23* is tasked with deploying yearly reports to your host. You want their deployed files to be placed only in the following locations:

*jupiter* – C:\reports\monthly

*114.342.23.23* – C:\reports\yearly

To enable this, you have assigned those allowed directories to the allowed hosts. Any attempt by *jupiter.mycompany.com* or *114.342.23.23* to deploy their files to directories other than the ones assigned to them in the `allowedDirectories` element with fail.

Figure 15 illustrates this example:

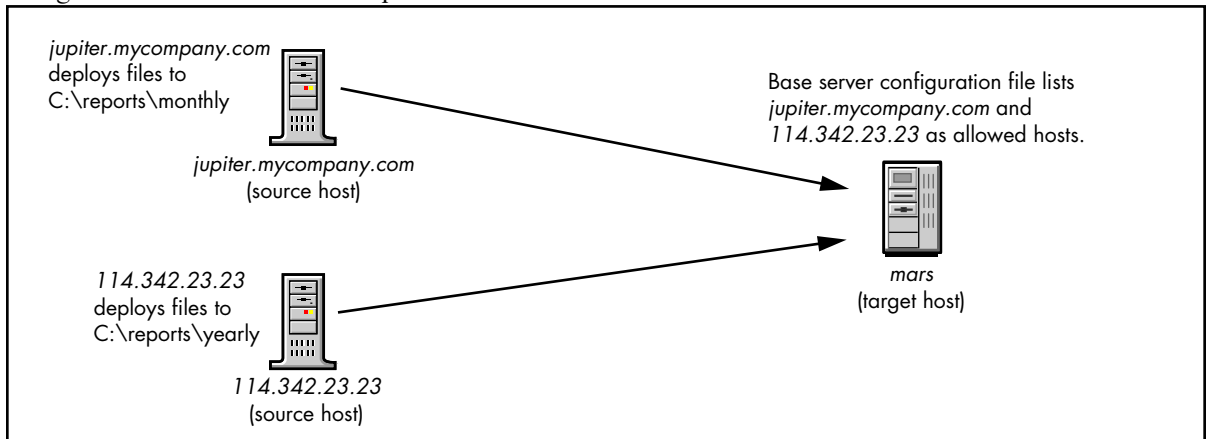


Figure 15: Allowed Hosts and Allowed Directories

## Logging

Each OpenDeploy base server host logs its activities to its associated base server log. This log contains entries on activities that occur regarding the base server host. See “Logging” on page 155 for more information on logging types and functions.

You can specify default logging values for all deployments that your OpenDeploy base server host performs or receives. If a particular deployment configuration has defined its own logging settings, the deployment-based logging settings supersede the default settings you made in the base server configuration file.

Default base server host logging is defined in the `logRules` element:

```
<deployServerConfiguration>
  ...
  <logRules ... />
</deployServerConfiguration>
```



The `logRules` element contains the following attributes:

- `maxBytes` — specifies the maximum size in bytes a log file is allowed to grow before the file is closed and OpenDeploy begins writing to a new file. This value is known as the *rollover threshold*. You can specify different byte measurements in the value, including megabytes (mb), kilobytes (kb), and bytes (b). For example:

```
maxBytes="50mb" or
```

```
maxBytes="50000kb" or
```

```
maxBytes="50000000b"
```

indicates that the log file size can grow to 50 megabytes before OpenDeploy will close that log file and start a new one.

Ensure that you include the proper measurement indicator when setting the `maxBytes` attribute value. If no recognizable size measurement is indicated, OpenDeploy will use the default, bytes (b).

- `directory` — specifies the absolute directory location for the log files. The default location is:

```
od-home/log
```

You can only specify the log directory in the base server or receiver configuration. Unlike the other attributes, you cannot override this setting in a deployment configuration.

- `level` — indicates the level and type of logging OpenDeploy will perform.
  - `verbose` — logs high level of detail on deployment events as they occur. This logging level is best suited for troubleshooting deployment problems or evaluating deployment performance. Verbose logging can create very large log files. This is the default logging level.
  - `normal` — logs standard status and error messages. In most cases, this level of logging provides sufficient detail to meet your needs.

## Default Logging Settings

The default base server logging configuration is:

```
<logRules
  maxBytes="32mb"
  directory="od-home/log"
  level="verbose"
/>
```

which specifies the following:

- The threshold size (32 MB) a log file can grow before it is rolled over and a new log file is opened.
- The log files (base server, receiver, deployment macro, and deployment micro) will reside in the log subdirectory of the value you indicate for *od-home*.
- The logging level is *verbose*.

You can change any of these values, but they will not become effective until the host is refreshed or the OpenDeploy services or daemons are restarted.

The inclusion of the `logRules` element and its attributes and their values provide default logging settings for any deployment that does not specify logging settings in its own configuration.

## Encryption

Your source host can encrypt deployed files using the following methods:

- Weak (40-bit) symmetric using key file-based encryption
- Strong (up to 168-bit) asymmetric key encryption using Secure Sockets Layer-based (SSL) encryption

These types of encryption cannot be used in conjunction with one another.

Encryption settings are defined in the `localNode` element in the base server configuration file. See “Encryption” on page 280 for more information.

## Deployment Job Queuing

OpenDeploy allows you to line up, or *queue*, a second occurrence of the same deployment, or *job*, to the host running the deployment. The OpenDeploy sending host will only queue the last same-named job, replacing an existing same-named job that's already on the queue, but not currently running. This action protects against concurrency problems that would arise if multiple instances of a deployment were sent simultaneously. This feature is useful in an environment where incremental changes are pushed out to production frequently, and there is a risk that a deployment might be submitted while a previous deployment having the same name is still running.

For example, if the OpenDeploy host mars started the deployment *reports*, and that same deployment was triggered again, perhaps by a different user, OpenDeploy will queue this second instance of *reports*, and deploy it after the initial instance has completed its run.

Deployment job queuing is enabled in the `initiatorProperties` element residing in the base server configuration file (by default `odbase.xml`). Give the `pendSession` attribute the value of `yes`. For example:

```
<deployServerConfiguration>
...
<initiatorProperties pendSession="yes" />
...
</deployServerConfiguration>
```

By default, deployment job queuing is not enabled, and the `pendSession` attribute value is `no`. If deployment job queuing is not enabled, any subsequent attempt to queue a job that is also currently running will result in the immediate rejection or failure of the new job attempting to be initiated. However, the existing job that is already running is not affected.

### Limitations

Deployment job queuing will not work for deployments whose configurations are identical, but have different names. Deployment job queuing can only apply to like-named deployments.

File list deployments can be queued, but OpenDeploy will not ensure that the files referenced in each instance are the same. Queuing deployments where the file list changes dynamically runs the risk of having a different set of files deployed between the first and second instance.

Deployment job queuing does not consider parameter substitution in the deployment configuration. OpenDeploy will queue jobs for a deployment using parameter substitution, even if the substituted values differ between jobs. See “Parameter Substitution” on page 265 for more information on this feature.

## Deploying to OpenDeploy Downward Revision Targets

OpenDeploy can deploy files to hosts with the OpenDeploy 4.5.2 receiver software installed if the OpenDeploy 4.5.2 sending (client) software is installed on the source host and the necessary configurations are made.

### Server Configuration

You must add the `oldOdHome` element within the `deployServerConfiguration` element of the base server configuration file (by default `odbase.xml`). Within the `oldOdHome` element you must specify the path to the sending software in the `path` attribute:

```
<deployServerConfiguration>
  ...
  <oldOdHome path="od452-home" />
  ...
</deployServerConfiguration>
```

where *od452-home* is the full path to where the OpenDeploy 4.5.2 sending software resides on the source host. For example:

```
<oldOdHome path="C:\Program Files\Interwoven\OpenDeploy" />
```

### Deployment Configuration

Additionally, you must add the `downRev` attribute to the `execDeploymentTask` element in each deployment configuration that will deploy files to an OpenDeploy 4.5.2 receiver. For example:

```
<deployment ...>
  <execDeploymentTask useDefinition="MYDEFINITION" downRev="4.5.2">
</deployment>
```

If your deployment include than one definition, you must include the `downRev` attribute in the `execDeploymentTask` element that corresponds to the appropriate definition.

## Modifying the Receiver Configuration File

When the receiver software is installed on a target host server, it includes the *receiver configuration file*. The receiver configuration file is an XML file derived from the deploy server DTD, the same as the base server configuration file. These two files are essentially the same, with the base server configuration file having some additional elements to address functions unique to the base server software. The receiver configuration file contains the elements and attributes for settings such as the following:

- Restrictions on what OpenDeploy features can be utilized on deployments originating from the sending server
- Ports
- Logging defaults
- Encryption settings

The name and location of the default receiver configuration file installed with the OpenDeploy receiver software is:

```
od-home/etc/odrcvr.xml
```

This file can be renamed. However, that name must be referenced in the OpenDeploy service configuration file (`deploy.cfg`) as the value of the `Deploy.serverNodesConfig` attribute. See “Modifying the Service Configuration File” on page 65 for more information.

The receiver configuration file serves the same purpose for receive-only target hosts that the base server configuration file does for source hosts that can both send and receive. Because both files are derived from the deploy server DTD, they share the same elements and attributes. However, in most cases, the receiver server configuration file contains fewer elements and attributes, given the reduced capabilities of a receive-only target host with the receiver software installed.

Like the base server configuration file, after the receiver server configuration file is configured, you do not have to modify it again unless there are changes to your host server or network. For a list of all elements and attributes that are required, or options within the receiver server configuration file, refer to the *OpenDeploy Reference*. For a basic discussion of XML structure and syntax, see “Understanding the Configuration DTDs” on page 195.

## Encoding

The encoding for the nodes configuration file can be encoding other than UTF-8. For example, if a value in the file contains Japanese characters, the encoding will need to be:

```
<? xml version="1.0" encoding="SHIFT_JIS" ?>
```

For French and German, the encoding value would be:

```
<? xml version="1.0" encoding="ISO-8859-1" ?>
```

Check what the appropriate value is for any non-ASCII characters and modify the nodes configuration file encoding as needed. If no encoding is specified, UTF-8 will be used by default.

## Communicating with Other Hosts

Receive-only target hosts use the bind port to communicate with source host, including participating in file comparisons and receiving deployed files. See “Communicating with Source and Target Hosts” on page 76 for more information.

## Logging

All target hosts log received deployments, whether the target is a receiver-only host or a source host. You can configure the target host’s logging in the receiver configuration file. The same logging configuration rules that apply to the base server host also apply to receiver hosts. The receiver configuration also includes the same default logging settings as the base server configuration. See “Logging” on page 85 for more information.

## File Transport Buffer Size

You can specify a buffer size for files being received in a deployment. See “Setting the File Transport Buffer Size” on page 77 for more information.

## Specifying Allowed Hosts

The ability to specify which source hosts can deploy files to your receive-only target host is configured in the same manner as for source hosts that can both send and receive. See “Specifying Allowed Hosts for Received Deployments” on page 83 for more information.

## Specifying Allowed Directories

The ability to specify the particular directory on your receiver-only target host that can receive deployed files from a source host is configured in the same manner as for source hosts that can both send and receive. See “Specifying Allowed Directories for Deployments” on page 84 for more information.

## Encryption

A receive-only target host can utilize encryption the same as a base server host can:

- Weak (40-bit) symmetric using key file-based encryption
- Strong (up to 168-bit) asymmetric key encryption using Secure Sockets Layer-based (SSL) encryption

These types of encryption cannot be used in conjunction with one another.

Encryption settings are defined in the `localNode` element in the base server configuration file. See “Encryption” on page 280 for more information.

## Using Different Administration Server Software

OpenDeploy requires a Tomcat server to be running for the OpenDeploy user interface to function properly. OpenDeploy includes a Tomcat server as part of the default installation for the administration server. OpenDeploy only has been tested with the Tomcat server that is installed as part of its installation package. Therefore, this is the only configuration of the Tomcat server that OpenDeploy officially supports. You can elect to obtain and use a different Tomcat server, providing that it contains the JDK 1.3 software. However, using OpenDeploy in conjunction with another Tomcat server is done so at your own risk.

To use a different version of the Tomcat server, or one residing in another location, modify the Tomcat configuration file accordingly. The Tomcat configuration file must point to the JDK 1.3 software for the OpenDeploy user interface to work. When you install the administration server software component, the JDK 1.3 software is included with it in the following location:

```
admin-home/servletd/java1.3
```

You can support OpenDeploy with your own Tomcat server by adding the OpenDeploy file `opendeploy.war` to your Tomcat server's `webapps` directory and restarting the Administrator service. You can find that `opendeploy.war` file in the following location:

```
od-home/httpd/webapps/opendeploy.war
```

## Windows

On Windows, the Tomcat configuration file resides at the following location:

```
admin-home\servletd\bin\tomcat.bat
```

To use JDK software that resides in another location, modify the `tomcat.bat` file to point to this location using the following syntax:

```
set JAVA_HOME=jdk-home
```

where *jdk-home* is the location where the JDK software resides. For example, if you wanted to point to the JDK 1.3 software residing at the following location:

```
c:\jdk1.3.0_2
```

then you would enter the following line into the `tomcat.bat` file:

```
set JAVA_HOME=c:\jdk1.3.0_2
```

Enter this line near the beginning of the Tomcat configuration file, immediately after the opening comments section.

The path to the JDK 1.3 software cannot contain any spaces. For best results, do not place it in such a location. If you cannot avoid pointing to a path with spaces, you can use the MS-DOS version of the path, which includes truncated directory names that utilize the tilde (“~”) symbol. For example, the path:

```
c:\Program Files\jdk1.3.0_02
```

would be entered as:

```
c:\PROGRA~1\jdk1.3.0_02
```



You can determine the truncated name using the following procedure:

1. Open a Command Prompt window.
2. Navigate to the directory one level above the directory whose name contains a space.
3. Enter the following command at the prompt:

```
dir /X
```

The contents of the directory is displayed, including the truncated name utilizing the tilde (“~”) symbol. That is the version of the directory name you must include in your JAVA\_HOME path.

## UNIX

The Tomcat configuration file for UNIX resides at the following location:

```
admin-home/servletd/bin/tomcat.sh
```

If you want to use JDK software that resides in another location, you must modify the `tomcat.sh` file to point to this location using the following syntax:

```
JAVA_HOME=jdk-home
```

where *jdk-home* is the location where the JDK 1.3 software resides. For example:

```
JAVA_HOME=/usr/local/jdk1.3
```

Enter this line near the beginning of the Tomcat configuration file, immediately after the opening comments section.

## Refreshing the opendeploy.war File

You might need to manually replace the `opendeploy.war` file located in the *admin-home* directory under the following circumstances:

- A new `opendeploy.war` file is issued as part of a patch.
- Files in the *admin-home/httpd/iwebapps/opendeploy* directory are lost or damaged and must be replaced.

In these situations, you must manually place the new or existing `opendeploy.war` file into its location within *admin-home* and expand it.

To refresh the `opendeploy.war` file, follow these steps:

1. Copy the `opendeploy.war` file, either from the patch (if appropriate), or from its location in *od-home*:

```
od-home/httpd/webapps
```

2. Place the copy of the `opendeploy.war` file in the following location in *admin-home*:

```
admin-home/httpd/iwebapps
```

3. Create the subdirectory `opendeploy` within that directory:

```
admin-home/httpd/iwebapps/opendeploy
```

4. Navigate to the `opendeploy` directory you just created, and expand the `opendeploy.war` file by entering the following command at the prompt:

```
admin-home/servletd/java1.3/bin/jar -xvf ../opendeploy.war
```

## Internationalization

Use the information in this section if you are running OpenDeploy 5.5.1 on a localized operating system.

### Service Configuration File Format

The service configuration file (`deploy.cfg`) must be in ASCII text. Non-ASCII characters are not allowed.

### Encoding for XML-Based Configuration Files

The following XML-based host configuration files can use encoding in formats other than UTF-8.

- Base server configuration file (by default `odbase.xml`)
- Receiver configuration file (by default `odrcvr.xml`)
- Nodes configuration file (by default `odnodes.xml`)
- All deployment configuration files

For example, if a value in the file contains Japanese characters, the encoding will need to be:

```
<? xml version="1.0" encoding="SHIFT_JIS" ?>
```

For French and German, the encoding value would be:

```
<? xml version="1.0" encoding="ISO-8859-1" ?>
```

Check what the appropriate value is for any non-ASCII characters and modify the nodes configuration file encoding as needed. If no encoding is specified, UTF-8 will be used by default.

## OpenDeploy-DataDeploy Integration

This integration solution for OpenDeploy and DataDeploy enables the deployment of file assets and database assets in one single transactional deployment. Refer to the *OpenDeploy Release Notes* for a listing of which releases of DataDeploy are supported by this release of OpenDeploy.

### Component Location

The solution components reside in the following location on the OpenDeploy host:

```
od-home/solutions/ddsync
```

### Setup

To set up the OpenDeploy-DataDeploy integration, follow these steps:

1. Move the example components to their respective locations described in “Component Descriptions” on page 99.
2. Install the `IWXML.pm` module for use with `iwperl` on the target production server. You can perform this task by copying the following file

```
od-home/solutions/perl/IWXML.pm
```

to any one of the @INC paths of `iwperl`. You can view the INC paths by entering the following command at the prompt:

```
iwperl -V
```

You must also modify `ddsync.ip1` on the target production server with the DCR types. See “Component Descriptions” on page 99 for more information.

3. (For internationalization support only) Install the `I18N_utils.pm` module from the following location:

```
od-home/solutions/perl/I18N.pm
```

into `iwperl` through a TeamSite subdirectory under one of the `iwperl @INC` paths. For example, to setup DataDeploy's `iwperl`, enter the following command at the prompt:

```
cp od-home/solutions/perl/I18N_utils.pm dd-home/iw-perl/lib/perl5/site_perl/5.005/TeamSite
```

If your `iwperl` was installed as part of TeamSite 5.5 or later, this module will already reside there. You must also copy the following file:

```
od-home/solutions/ddsynch/odxmlenc.map
```

to your `od-home` location. This mapping file specifies the encoding you want to be used by the OpenDeploy internal XML log header.

4. Modify the source and location paths as needed:

- For a full deploy: `oddd_full.xml`
- For a delta deploy: `oddd_delta.xml`

You also can take advantage of the OpenDeploy parameter substitution feature for more dynamic specifications.

5. Regenerate by entering one of the following commands at the prompt. Different syntax is used depending on whether wide-table format or user-defined schema support is desired.

- Wide-table format:

```
iwsyncdb.ipl -genloadcfg loaddb.cfg workarea
```

- User-defined schema:

```
iwsyncdb.ipl -genloadcfg outfile-name -dumpdir fullpath-production-dumpdir -targetdir fullpath-production-targetdir development-workarea
```

6. Manually change `loaddb.cfg` to reflect the following:

- The path location to `database.xml`
- Any changes to templating types schemas

## Component Descriptions

This section contains the path and name of the OpenDeploy and DataDeploy integration components, and their descriptions. Note that *dd-home* refers to the DataDeploy home directory and *od-home* refers to the OpenDeploy home directory for the base server and receiver software.

- *dd-home/bin/ddsync.ipl* — the DNR script. This file is installed on both development and production servers. For user-defined schema support, customization is required. The DCR types must be specified. In *ddsync.ipl*, search for:

USER CUSTOMIZATION REQUIRED

For internationalization support, your host's local encoding must be specified. This field is also found under the CUSTOMIZATION section. The default value is NONE.

- *dd-home/conf/database.xml* — the “include” file for the database element. This file is installed on a production server.
- *dd-home/conf/subxmldb.template* — a file used by *iwsyncdb.ipl -genloadcfg* to generate a `loaddb.cfg`. This file is installed on the development server to generate `loaddb.cfg`, which then needs to be relocated to a production server.
- *dd-home/conf/subxmldb\_uds.template* — see description for *dd-home/conf/subxmldb.template* above.
- *dd-home/conf/subxmldb\_uds\_custom.template* — see description for *dd-home/conf/subxmldb.template* above.
- *dd-home/conf/loaddb.cfg* — a generated DataDeploy configuration file used to run DataDeploy to update the database from dump files. This file is generated on the development server, but needs to be relocated to a production server. This file can be regenerated through the following command:

***iwsyncdb.ipl -genloadcfg output-path/loaddb.cfg -dumpdir production-dump-dir targetdir production-target-dir area-vpath***



- *dd-home/conf/tsxml.cfg* — a DataDeploy configuration file used to create the XML dump files from TeamSite. This file is installed on the development server.
  - Using wide-table format — copy from *\*/tsxml.cfg.example-wide*
  - Using user-defined-schema — copy from *\*/tsxml.cfg.example-uds*
- *od-home/conf/odrcvr.xml* — the configuration file for an OpenDeploy host with the receiver software installed. This type of OpenDeploy host can only receive files. This file is installed on a production server.
- *od-home/conf/oddd\_full.xml* — an OpenDeploy source host configuration file. This file is installed on the development server.
- *od-home/conf/oddd\_delta.xml* — an OpenDeploy source host configuration file. This file is installed on development machine.
- *od-home/odxmlenc.map* — an OpenDeploy mapping file. This mapping file specifies encoding to use in the OpenDeploy internal XML log.

## Usage

This section describes how these integration tools are used.

- On a development server, run the following command at the prompt:

```
iwodstart oddd_full or  
iwodstart oddd_delta
```

If you want to use parameter substitution, enter the following command at the prompt:

```
iwodstart oddd_full -k "var1=value1"
```

where the *\$var1* parameter is present somewhere in the *oddd\_full.xml* configuration file.

- On a production server, start up the OpenDeploy receiver services or daemons.

## How the Integration Works

This section describes the processes that OpenDeploy and DataDeploy perform to complete the integration.

1. OpenDeploy runs the specified deployment transactionally.
2. A pre-deploy Deploy and Run script is kicked off to invoke DataDeploy.
  - The script (`ddsync.ip1`) is a wrapper that invokes DataDeploy to extract TeamSite DCR or TeamSite metadata into XML dump files.
  - The configuration files for the DataDeploy invocation have been prepped to dump out the DCR types and metadata of a specified area.
3. OpenDeploy transfers its normal set of files plus the XML dump files.
4. A post-deploy Deploy and Run script is kicked off to invoke DataDeploy:
  - The `ddsync.ip1` script invokes DataDeploy to import into the configured database all the actual DCR files. No transient XML dump file is used here.
  - The config files for the DataDeploy invocation have been prepped to read a series of generated file lists of the DCR types. Metadata is the exception and still follows the wide-table scheme. The generated file lists are derived from parsing the OpenDeploy internal XML log. For internationalization, these generated file lists are in the UTF-8 encoding for DataDeploy usage.
5. Both file assets and database assets should now be deployed. If any part of the deploy should fail, the entire deployment should be restored back to previous state. For example:
  - If the DataDeploy component fails, the OpenDeploy deployed files will be reverted.
  - If the OpenDeploy component fails, the DataDeploy component will not be invoked.

## ddsync.ipl Usage

Here are the different ways `ddsync.ipl` can be used:

### Wide-Table Format:

```
ddsync.ipl area_top dump_dir dump full area
```

```
ddsync.ipl area_top dump_dir dump differential area basearea
```

```
ddsync.ipl area_top dump_dir load full
```

```
ddsync.ipl area_top dump_dir load differential
```

### User-Defined Schema:

```
ddsync.ipl area_top dump_dir dumpea full area
```

```
ddsync.ipl area_top dump_dir dumpea differential area basearea
```

```
ddsync.ipl area_top dump_dir loaduds full
```

```
ddsync.ipl area_top dump_dir loaduds differential
```

<i>area_top</i>	The absolute path to top of area directory.
<i>dump_dir</i>	The relative path to dump directory in area.
dump	Dump TeamSite metadata to file.
load	Load database from dump file.
load_uds	Load database from DCR filelist.
full	The entire area traversal.
differential	Comparing between two areas.
<i>basearea</i>	If differential is used, the <i>basearea</i> is the previous area.
<i>area</i>	If differential is used, the <i>area</i> is the current area.

The `ddsync.ipl` usage also creates the following log file:

`dd-log-home/ddsync_dump_load.log`

## Notes

- Run only directory comparison deployments. TeamSite-based and filelist modes are not supported at this time.
- OpenDeploy-DataDeploy integration is most suitable for deployment of editions.
- Because OpenDeploy uses remote file system comparison and DataDeploy is either full or differential, the consistency of the files and their respective database data is not tightly maintained. The deployment workflow process must be tightly adhered to. For example, the first development-to-production deployment of an edition will deploy all files and all database assets. Subsequent deployments will be directory comparisons between a subsequent edition and the production server and tuple-diffs between current edition and the previous edition. Injection of new files or database content on any point outside of the edition deployment process will create data inconsistencies.
- Be sure to change all references to `/local/iw-home` in `ddsync.ipl` and `oddd_send.cfg` to reflect the location of the actual OpenDeploy home directory. The `oddd_send.cfg` and `oddd_receive.cfg` files are OpenDeploy sender and receiver host configuration files, and as such require specific configuration for both types of hosts. If these configuration files already exist on systems targeted for the integration, the sample configuration files should be integrated into the production systems.

## Performing Authentication Through a Firewall

If an OpenDeploy host is deploying content through a firewall, you must configure your OpenDeploy environment in the following manner:

- The port number you assigned the OpenDeploy host (by default port 20014) must be open on the firewall.
- The nodes configuration file (by default `odnodes.xml`) on the source host must contain the target host's IP address.
- The deployment configuration file must contain the firewall's IP address as the `localNode` element's `host` attribute value.
- The base server or receiver configuration file of the target host (by default `odbase.xml` or `odrcvr.xml`) must contain the target host's IP address as the `localNode` element's `host` attribute value.
- The base server or receiver configuration file of the target host must also include the firewall's IP address as an allowed host. Within the `allowedHost` element, you must assign the firewall's IP address as the `node` element's `host` attribute value. See "Specifying Allowed Hosts for Received Deployments" on page 83 for more information.

## Uninstalling OpenDeploy

Uninstall OpenDeploy software components one at a time, similar to the way you installed them. In some cases you will want to remove all of the components on a server. In other cases you might only want to remove some components, such as if you have a single-host installation of all the components on a single server and you want to spread them out over several servers. Uninstallation of OpenDeploy software varies depending on the server platform.

### Windows

To uninstall each OpenDeploy software component, you must perform the following tasks in order:

- Stop the OpenDeploy service for each corresponding software component.
- Uninstall the software component itself, using the Windows Add/Remove Programs tool.
- Remove any remaining files and directories as necessary. Even after uninstallation, OpenDeploy leaves some legacy files and directories, such as *od-home/conf*. This allows you to retain files such as customized configuration files that you might want to maintain even if you decide to reinstall OpenDeploy.

#### Stopping the OpenDeploy Services

See “Stopping OpenDeploy” on page 113 for instructions on which OpenDeploy services are associated with which software components, and how to stop those services. After you have stopped the appropriate OpenDeploy services, you can now uninstall the OpenDeploy software.

#### Uninstalling the OpenDeploy Software

To uninstall the OpenDeploy software on a Windows server, follow these steps:

1. Open the Add/Remove Programs window. This process may differ depending on the version of Windows you are using.

2. Select the OpenDeploy software component you want to uninstall in the **Currently Installed Programs** list:
  - Interwoven AdminServer
  - Interwoven OpenDeploy Base Sender (sending hosts)
  - Interwoven OpenDeploy Receiver (receive-only hosts)
  - Interwoven Operation Server
3. Click **Change/Remove** to remove the software.
4. Repeat this procedure for the remaining OpenDeploy software components you want to remove.

Certain files and directories remain on your host after the uninstallation, such as log files and configuration files. This allows you to keep a record of your OpenDeploy activities even after the software is removed. If you want to remove these files as well, you can manually delete them from within the Windows Explorer.

## UNIX

To uninstall each OpenDeploy software component, you must perform the following tasks in order:

- Stop the OpenDeploy daemon for each corresponding software component.
- Delete the appropriate OpenDeploy files and directories. You might want to keep log and configuration files for later use.

### Stopping the OpenDeploy Daemons

See “Stopping OpenDeploy” on page 113 for instructions on which OpenDeploy daemons are associated with which software components, and how to stop those daemons. After you have stopped the appropriate OpenDeploy services, you can now uninstall the OpenDeploy software.

### Uninstalling the Operations Server Software

To uninstall the administration server software on a UNIX server, follow these steps:

1. Navigate to the following location:

```
ops-home/OpenAPI
```

2. Run the uninstallation script to remove the operations server software by entering the following command at the prompt:

```
./uninstallopserver
```

### Uninstalling the Administration Server Software

To uninstall the administration server on a UNIX server, follow these steps:

1. Navigate to the following location:

```
admin-home/AdminServer
```

2. Run the uninstallation script to remove the administration server software by entering the following command at the prompt:

```
./uninstalladmin
```

### Uninstalling the Base Server and Receiver Software

To uninstall the base server or receiver software on a UNIX server, follow these steps:

1. Navigate to the following location:

```
od-home/install
```

2. Run the uninstallation script to remove the OpenDeploy software by entering the following command at the prompt:

```
./uninstallod
```



## Chapter 3

# Getting Started

---

This chapter introduces the basic features and functions of OpenDeploy that you can perform from the user interface and command line. For some users, this information is sufficient to meet their OpenDeploy needs. For others, particularly those who want to create more complex deployment configurations, this chapter will provide the foundation upon which they can go on to the more advanced features described later in this book.

## Starting OpenDeploy

OpenDeploy is run by starting its services or daemons on the host server. This process differs depending on the type of platform on which it is operating.

### Windows

Start OpenDeploy services on a Windows server by using one of the following methods:

- Rebooting
- Starting the OpenDeploy services from the Services window
- Starting from the command line

#### Starting by Rebooting

After the OpenDeploy software is successfully installed on the server, the OpenDeploy services will automatically start upon rebooting. Be sure to have your bootstrap administrator already configured before rebooting. See “Configuring the Bootstrap Administrator” on page 67.

## Starting from the Services Window

You can start OpenDeploy services from the Services window. You might prefer this method if the OpenDeploy services were previously stopped and it is impractical to restart the server.

OpenDeploy services can vary depending on what software components you have installed. Here is a table of OpenDeploy services and their corresponding software components:

Service	Software
Interwoven OpenAPI	Operations server
Interwoven UI Admin	Administration server
Interwoven OpenDeploy Service	OpenDeploy Sender (sending host only) and Receiver (receive-only host only)

To start the OpenDeploy services, follow these steps:

1. Open the Services window. This process may differ depending on which version of Windows you are using.
2. Right-click on **Interwoven OpenAPI** and select **Start** from the pop-up menu.
3. Right-click on **Interwoven UI Admin** and select **Start** from the pop-up menu.
4. Right-click on the **Interwoven OpenDeploy Service** and select **Start** from the pop-up menu.

## Starting From the Command Line

To start one or more OpenDeploy services from the Windows command line, follow these steps:

1. Open a command line window and navigate to the following location:

```
od-home\bin
```

2. Enter the following command to start the Interwoven OpenAPI service:

```
net start "Interwoven OpenDeploy Service"
```

3. Enter the following command to start the Interwoven UI Admin:

```
net start "Interwoven UI Admin"
```

4. Enter the following command to start the Interwoven OpenDeploy service:

```
net start "Interwoven OpenDeploy Service"
```

## UNIX

Start OpenDeploy daemons on a UNIX server by using one of the following methods:

- Rebooting the server — After OpenDeploy software is successfully installed, OpenDeploy daemons will automatically start upon rebooting the server. Be sure to have your bootstrap administrator already configured before rebooting after installing OpenDeploy software on your host. See “Configuring the Bootstrap Administrator” on page 67.
- Entering the start command at the prompt — In some cases it is not practical to shut down the server to start the OpenDeploy daemons. You can start OpenDeploy without rebooting the server by navigating to the location of the OpenDeploy start program.

### Starting the Operations Server

To start the operations server that was installed with OpenDeploy, follow these steps:

1. Navigate to the following location:

```
ops-home/etc/init.d or  
/etc/init.d
```

2. Start the operations server by entering the following command at the prompt:

```
./opsstart start
```

### Starting TeamSite OpenAPI

If you are using OpenDeploy in conjunction with TeamSite 5.5 or later, you must use the TeamSite OpenAPI instead of the OpenDeploy operations server. See “Using TeamSite OpenAPI with OpenDeploy” on page 57 for more information.



### Starting the Administration Server

To start the administration server, follow these steps:

1. Navigate to the following location:

```
admin-home/servletd/bin or  
/etc/init.d
```

2. Start the administration server by entering the following command at the prompt:

```
./iwtcboot start
```

### Starting the Base Server and Receiver

To start the base server or receiver software, follow these steps:

1. Navigate to the following location:

```
od-home/etc/init.d or  
/etc/init.d
```

2. Start the OpenDeploy software by entering the following command at the prompt:

```
./iwod start
```

### Starting the User Interface

Starting OpenDeploy does not automatically start the OpenDeploy user interface. After you have OpenDeploy running, you can access the user interface using a supported Web browser. See “OpenDeploy User Interface” on page 119 for more information.

## Stopping OpenDeploy

OpenDeploy is quit by stopping its services or daemons on the host server. This process differs depending on the type of platform on which it is operating.

### Windows

You must stop the various OpenDeploy services running on your Windows server to stop or quit OpenDeploy. The following services correspond to the OpenDeploy software components:

Service	Software
Interwoven OpenAPI	Operations server
Interwoven UI Admin	Administration server
Interwoven OpenDeploy Service	OpenDeploy Sender (sending host only) and Receiver (receive-only host only)

To stop the OpenDeploy services, follow these steps:

1. Open the Services window. This process may differ depending on which version of Windows you are using.
2. Right-click on the **Interwoven OpenAPI** and select **Stop** from the pop-up menu to stop the operations server.

The Interwoven UI Admin service is tied to the Interwoven OpenAPI service. If you stop the Interwoven OpenAPI service, you will be notified that the Interwoven UI Admin service will also be stopped.

If you are using OpenDeploy in conjunction with TeamSite, stopping this service will also stop TeamSite.

3. (If necessary) Right-click on **Interwoven UI Admin** and select **Stop** from the pop-up menu to stop the administration server.
4. Right-click on the **Interwoven OpenDeploy Service** and select **Stop** from the pop-up menu to stop the base server or receiver software.

## UNIX

To stop the OpenDeploy daemons on a UNIX server, follow these steps:

### Stopping the Operations Server

To stop the operations server that was installed with OpenDeploy, follow these steps:

1. Navigate to the following location:

```
ops-home/iwopenapi or  
/etc/init.d
```

2. Stop the operations server by entering the following command at the prompt:

```
./opsstart stop
```

### Stopping TeamSite OpenAPI

Refer to your TeamSite documentation for instructions on how to stop TeamSite OpenAPI.

### Stopping the Administration Server

To stop the administration server, follow these steps:

1. Navigate to the following location:

```
admin-home/servletd/bin or  
/etc/init.d
```

2. Stop the administration server by entering the following command at the prompt:

```
./iwtcbboot stop
```

## Stopping the Base Server and Receiver

To stop the base server or receiver, follow these steps:

1. Navigate to the following location:

```
od-home/etc/init.d or  
  
/etc/init.d
```

2. Stop the base server or receiver software by entering the following command at the prompt:

```
./iwod stop
```

## Refreshing the OpenDeploy Server

Any time you modify certain elements in the base server, receiver, or nodes configuration files, you must reset your OpenDeploy base server or receiver service or daemon to accept the changes. You can refresh your OpenDeploy server without rebooting or manually restarting individual services or daemons by using the `iwodserverreset` command-line tool. The `iwodserverreset` command-line tool refreshes the OpenDeploy server with the new settings specified in the updated configuration files. Using this tool, you can make changes to the server configuration file and have it be read without having to bring down your services or restart your host.

The `iwodserverreset` command-line tool will not cause the configuration to be refreshed if there are deployments in progress at the time the command is run.

The following elements in the base server and receiver configuration files (by default `odbase.xml` and `odrcvr.xml`) can be refreshed using `iwodserverreset`:

- `allowedHosts`
- `allowedDirectories`
- `logRules`

The following elements and their attributes in base server and receiver configuration file are not refreshable:

- `localNode`
- `initiatorProperties`
- `listenerProperties`
- `transportProperties`
- `schedulerProperties`

To make the server read and implement changes in these elements, you must restart your OpenDeploy services or daemons.

Refer to the *OpenDeploy Reference* for descriptions of these elements and their attributes.

To refresh your OpenDeploy server's configurations, follow these steps:

1. Navigate to the following directory:

```
od-home/bin
```

2. Reset the OpenDeploy services by entering the following command at the prompt:

```
iwodserverreset
```

There are various options associated with the `iwodserverreset` command-line tool. Here is a listing of these options, along with the usage syntax:

```
iwodserverreset -h | -v | -q
```

<code>-h</code>	Displays usage information.
<code>-v</code>	Displays version information.
<code>-q</code>	Disables messages generated when there are active deployments in progress at the time <code>iwodserverreset</code> is run.

## Dependencies on the Operations Server

One of the underlying services of Interwoven products such as OpenDeploy and TeamSite 5.0 and later is OpenAPI. The operations server that comes with OpenDeploy is a subset of the OpenAPI. If you would like more information about OpenAPI, consult the administration chapter of the *OpenAPI Developer's Guide*, available on-line as part of the OpenAPI SDK.

### Administration Server

The operations server or OpenAPI server must be up and running before the user interface administration service can be started.

### Verifying OpenAPI Operation

To verify that this service has been successfully started after an installation, follow these steps.

#### Windows

The Services window should display “Interwoven OpenAPI,” running as “administrator,” with status “started.” If the OpenAPI service is not running, then it needs to be restarted.

#### UNIX

On UNIX, the following command should display the OpenAPI daemon running:

```
ps -ef | grep IWServiceServer
```

If the OpenAPI daemon is not running, then it needs to be started manually.

## **RMI Registry Service Considerations**

Both the operations server, including OpenAPI, and the base server make use of the RMI registry service. Interwoven products using this service can work together without a problem. However, other software products that make use of the RMI registry can cause conflicts that might prevent OpenDeploy and other Interwoven products from working properly. It is strongly recommended that no other programs accessing the same RMI registry as the ones used by OpenDeploy and TeamSite be included in your OpenDeploy environment.

The RMI registry ports being used are:

- Operations server (including OpenAPI) — 1099
- OpenDeploy — 9173 (default) or a port you specify during installation.

## **Installing TeamSite in an OpenDeploy Environment**

If you are installing TeamSite 5.0.1 or later into an environment where OpenDeploy already exists, you must uninstall the OpenDeploy operations server software prior to performing the TeamSite installation. Included in the TeamSite installation is OpenAPI. OpenDeploy can use OpenAPI in place of the operations server for access and role management. After installing TeamSite, you must reconfigure OpenDeploy to use OpenAPI. See “Using TeamSite OpenAPI with OpenDeploy” on page 57 for more information.

## OpenDeploy User Interface

The user interface enhances your ability to perform and monitor OpenDeploy tasks anywhere in the enterprise. All that is needed is a supported Web browser, which avoids the need to install additional client software on each computer from which you might want to administer the product. Refer to the *OpenDeploy Release Notes* for a list of supported browsers. You must also have the OpenDeploy administration server software or an alternate Tomcat server installed and its service or daemon running.

The user interface is a tool to help you learn the product and start using OpenDeploy right away. You will also find the graphical monitoring and scheduling features an advantage in managing deployments. However, more advanced features and configurations will still require you to work directly with configuration files and command-line tools.

If you are starting OpenDeploy for the first time after installing it, you must follow the procedures for first time start-up and login. After you have configured your server to recognize specific hosts, and have created the Administrator and User roles for individuals, you can follow the normal procedures for starting the OpenDeploy user interface.

### Browser Refresh Requirements

To ensure that changes you make to OpenDeploy are reflected in the user interface, you should configure your Web browser to refresh a page each time it is visited.

### Accessing the User Interface

To access the OpenDeploy user interface, point your browser to the following URL:

```
http://admin-server-hostname:admin-port-number/iw/opendeploy/login
```

where *admin-server-hostname* is the name of the host server running the administration server software, and *admin-port-number* is the administration server port number you chose when you installed the administration server software. For example, if your administration server host was named *andromeda* and you entered the administration port number 8081, then the URL to access the user interface would be:

```
http://andromeda:8081/iw/opendeploy/login
```

If you access the user interface from the same server on which the administration server software is running, you can use “localhost” as the administration server in the URL.

## Login

Starting the user interface displays the login window (Figure 16).

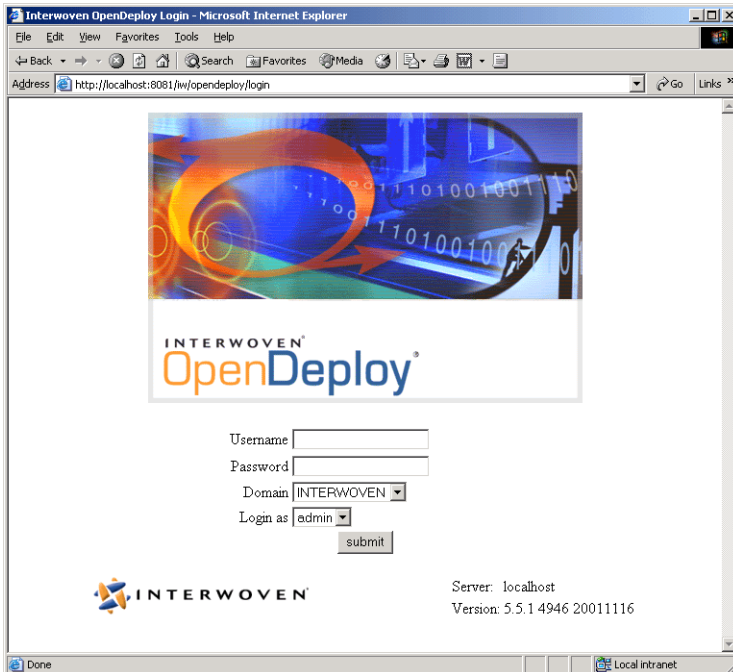


Figure 16: OpenDeploy Login Window on Windows Server

The standard method of logging into OpenDeploy requires entering the following information:

- User name
- Password
- Domain (Windows only)
- Role — either `admin` or `user`. See “Roles and Authorization” on page 132 for more information.

## First Time Login Using Bootstrap

When OpenDeploy is first installed on a server, you must log in as the bootstrap administrator to gain full administrator access. If you have not configured a bootstrap administrator yet, you must do so before you can log in. See “Configuring the Bootstrap Administrator” on page 67 for more information. After you have logged on as the bootstrap administrator, you need to verify and add your OpenDeploy server host to the list of OpenDeploy hosts that are accessible by the administration server. See “Adding OpenDeploy Hosts” on page 122 for more information.

## Timeout Setting

OpenDeploy includes a timeout feature for the user interface. This timeout feature is a security measure to prevent unauthorized access to an OpenDeploy user interface window that has been idle past the timeout period. Users who attempt to perform any task through the user interface past the timeout period will have to log in again. The default timeout period is measured in minutes. The default period is 9999 minutes. However, you can change this value to any amount you want.

To change the timeout value of the OpenDeploy user interface, follow these steps:

1. Open the following file using your favorite text or XML editor:

*admin-home/webapps/opendeploy/WEB-INF/web.xml*

2. Enter the amount of time in minutes that you want for the timeout value in the following location within the *web.xml* file:

```
<session-config>
  <session-timeout>xxxx</session-timeout>
</session-config>
```

where *xxxx* is the number of minutes you want. If you are changing this value for the first time, the existing value will be 9999.

3. Save your changes and close the *web.xml* file.
4. Restart your administration server service or daemon. See “Starting OpenDeploy” on page 109 for more information.

The OpenDeploy user interface automatically will timeout when idle past the number of minutes you entered.

## OpenDeploy Host Management

You can manage multiple OpenDeploy hosts from a single browser running the OpenDeploy user interface. You can apply OpenDeploy features and functionality available through the user interface to any OpenDeploy host listed, assuming you have the proper access and permissions, and the OpenDeploy host is capable of performing the particular task. You must add each OpenDeploy host you want to manage into the OpenDeploy Servers window (Figure 17). You can also modify and delete existing servers from this window.

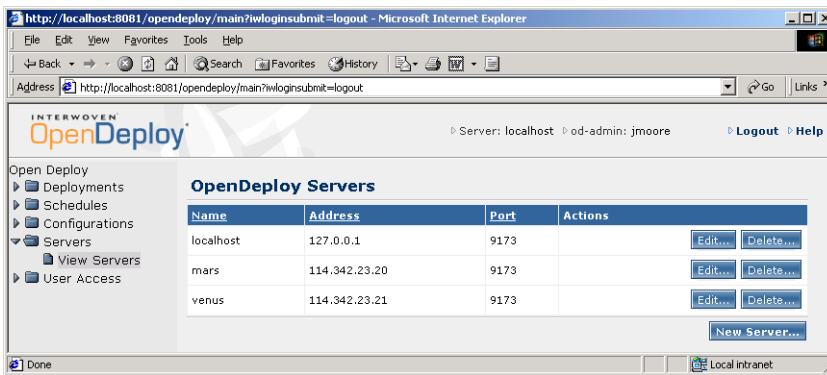


Figure 17: OpenDeploy Servers Window

## Adding OpenDeploy Hosts

To add an OpenDeploy host to your user interface, follow these steps:

1. Select **Servers > View Servers** to display the OpenDeploy Servers window. Here you can see which OpenDeploy hosts are currently listed, including their fully qualified DNS host name or IP address, and RMI registry port number.

2. Click **New Server** to display the New Server window (Figure 18).

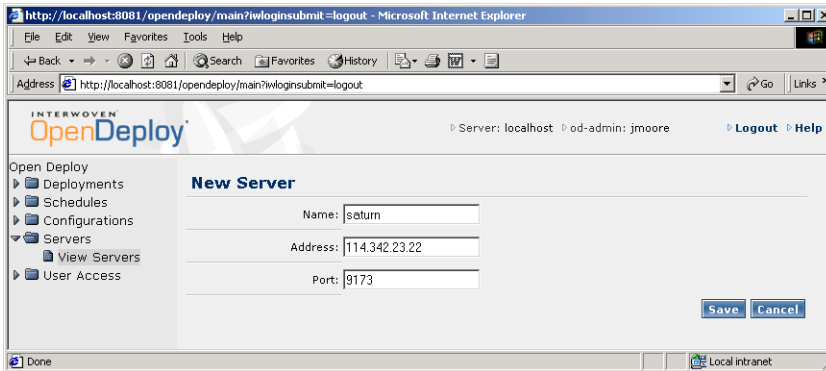


Figure 18: New Server Window

3. Input the following information regarding the new host you are adding into the new host template:
  - **Name** — the logical name you define for the host.
  - **Address** — the fully qualified DNS host name or IP address of the host.
  - **Port** — the port assigned to the RMI registry service.
4. Click **Save** to add the new server. The OpenDeploy Servers window reappears (Figure 19) with the added host.

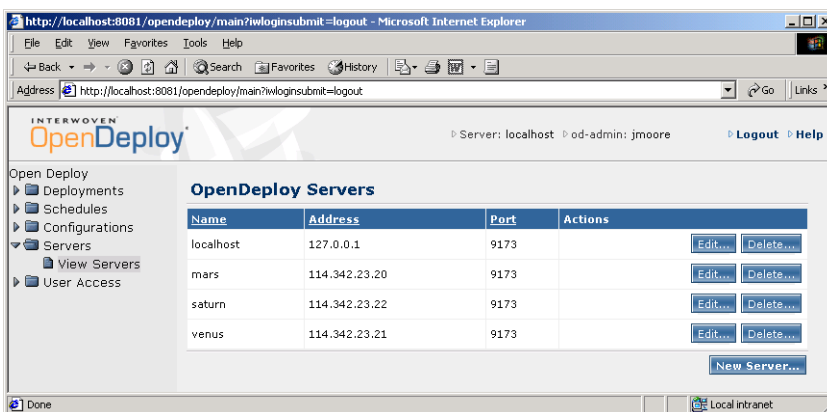


Figure 19: OpenDeploy Servers Window with Added Server

## Changing Server Information

You can change the name, IP address, and RMI registry service port for any OpenDeploy host listed in the OpenDeploy Servers window.

To change the information of a selected OpenDeploy host, follow these steps:

1. Select **Servers > View Servers** to display the OpenDeploy Servers window.
2. Click the **Edit** button that corresponds to the OpenDeploy host server whose information you want to modify. The Edit Server window (Figure 20) appears.

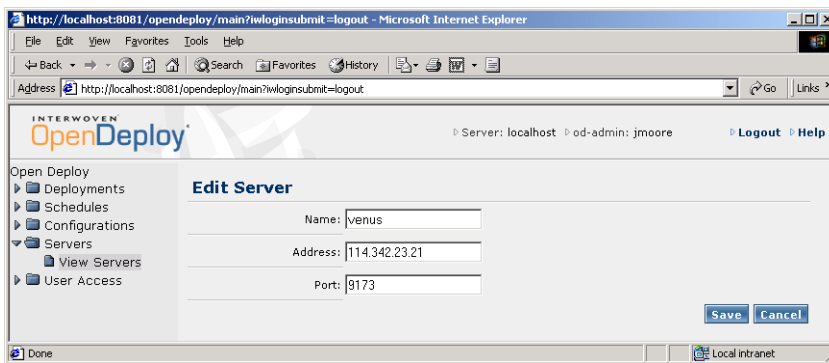


Figure 20: Edit Server Window

3. Make your modifications as necessary and click **Save**. The OpenDeploy Servers window reappears, reflecting the changes you made to the host.

## Deleting OpenDeploy Hosts

You delete hosts from the OpenDeploy user interface through the OpenDeploy Servers window. You are only deleting the listing of that server in the user interface, not deleting any OpenDeploy software from the actual server. You can add any server you have previously deleted at any time.

To delete a selected OpenDeploy host from the user interface, follow these steps:

1. Select **Servers > View Servers** to display the OpenDeploy Servers window (Figure 17).

2. Click the **Delete** button that corresponds to the OpenDeploy host server that you want to delete from the user interface.
3. Click **OK** when prompted to confirm that you want to delete that selected host. The OpenDeploy Servers window is refreshed, no longer displaying the server you just deleted.

## Monitoring Host Logs and Configurations

The Server Management window (Figure 21) provides a single location in the user interface where you can monitor and access the configuration and log files associated with a selected OpenDeploy host.

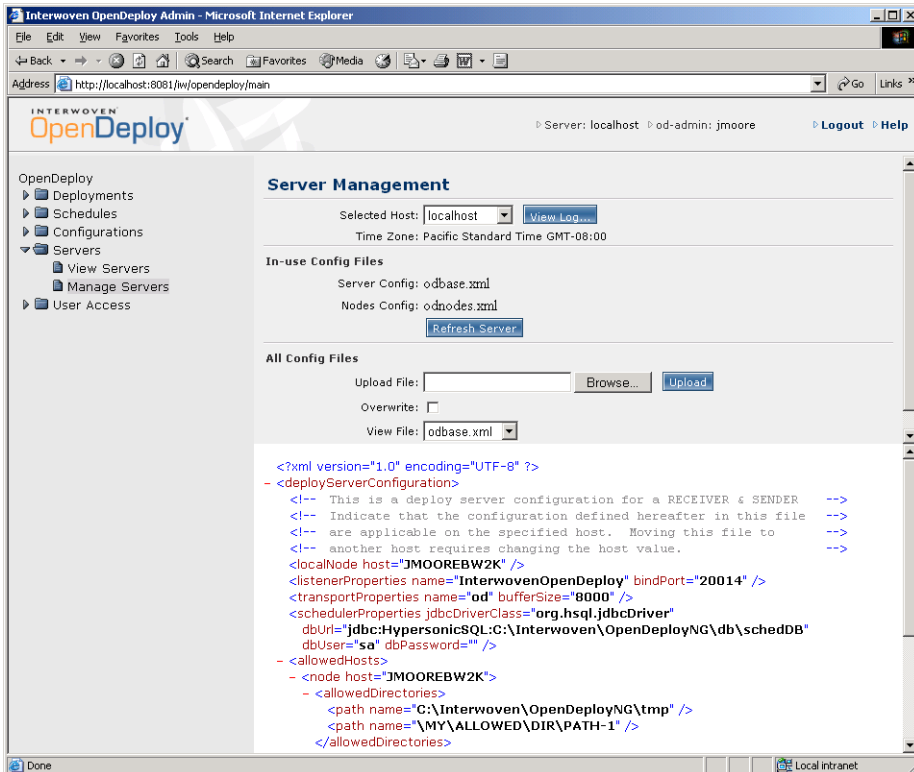


Figure 21: Server Management Window

Within the Server Management window you can perform the following host configuration and log file tasks:

- View the OpenDeploy base server and receiver log files of the selected OpenDeploy host.
- View the names of the host configuration files currently in use by the selected OpenDeploy host.
- Upload a host configuration file that you created or modified locally to a selected OpenDeploy host.
- Refresh the OpenDeploy host services and daemons to reread and implement changes in configuration.
- Display the XML code of a host configuration file located in the *od-home/etc* directory of a selected host.

### Viewing the Base Server and Receiver Log Files

You can access and view the base server and receiver log files from the Server Management window. Refer to the *OpenDeploy Administration Guide* for information on the contents of the base server and receiver log files.

To view the base server and receiver log files, follow these steps:

1. Select **Servers > Manage Servers** to display the Server Management window.
2. Select the name of the OpenDeploy server whose base server or receiver log you want to view from the **Selected Host** list.
3. Click **View Log**. A separate browser window appears displaying the OpenDeploy Log Viewer window containing the base server or receiver log file of the host you chose.

## Uploading Host Configuration Files

The Server Management window provides you the ability to upload host configuration files (base server, receiver, and nodes configuration files) from your local computer into the *od-home/etc* directory of the selected host. Any file you upload must be a valid XML file that follows the deploy server or nodes configuration DTD.

For example, if you wanted to add or modify the base server file of a particular OpenDeploy host, you could create or make the appropriate changes to that file and upload it to that host. You must still use a text or XML editor to modify the configuration file. You cannot modify the file from the Server Management window, only copy it to the host. This feature is only available to individuals holding an Administrator role on the affected OpenDeploy host. See “Roles and Authorization” on page 132 for more information.

Any modified configuration file you upload must have the same file name as the one you intend to replace. Names of the configuration files currently in use by the selected OpenDeploy host are displayed in the **In-use Config Files** section of the Server Management window. You can also upload other files with the *.xml* extension, but the OpenDeploy host will not be able to read those files upon refreshing. Additional steps are required to substitute a differently named host configuration file for one currently in use by an OpenDeploy host.

If you want to modify a host configuration file currently in use by the selected OpenDeploy host, you are responsible for obtaining a copy of that file and placing it on your local computer. You cannot download host configuration files through the OpenDeploy user interface. You will have to map a drive to the selected host or to find another method to copy the file you want from the selected host to your local computer.

To upload a host configuration file to a remote OpenDeploy host, follow these steps:

1. Create a new host configuration file or modify an existing one on your local computer.
2. Select **Servers > Manage Servers** to display the Server Management window.
3. Select the name of the OpenDeploy server to which you want to upload files from the **Selected Host** list.

4. Enter the path to the configuration file you want to upload to the selected host in the **Upload File** box. You can also click **Browse** and navigate to the location of the file. Any file you upload must have the `.xml` extension.
5. Check the **Overwrite** box. This is required any time you are overwriting an existing host file.
6. Click **Upload** to copy the file you specified from your computer to the selected OpenDeploy host. Your file will be copied to the `od-home/etc` directory, and will overwrite the existing file there.

The OpenDeploy host receiving your uploaded file will not run with the changes in the updated configuration file until you refresh the server.

### Viewing Host Configuration Files

You can view the XML source code of the host configuration files, and other XML-based files that reside in the `od-home/etc` directory of a selected OpenDeploy host, through the Server Management window. This is a quick and convenient way to see how a selected OpenDeploy host is configured. Only valid XML files will be displayed. Any other file will produce an error message, even if it appears in the **View File** list. This feature is only available to individuals holding an Administrator role on the selected OpenDeploy host. See “Roles and Authorization” on page 132 for more information.

To view the source code of an OpenDeploy host configuration file, follow these steps:

1. Select **Servers > Manage Servers** to display the Server Management window.
2. Select the name of the OpenDeploy server whose host configuration file source code you want to view from the **Selected Host** list.
3. Select the file whose source code you want to view from the **View File** list. The source code is displayed in the **Configuration** window.

If a file you want to view does not appear in the **View File** list, ensure that the file resides in the `od-home/etc` directory of the selected host. If you receive an error message after selecting a file, it may not be a properly formatted XML file.

## Refreshing the OpenDeploy Host Server

You can refresh the selected OpenDeploy host through the Server Management window. Refreshing the host causes it to reread its host configuration files currently in use. These files are displayed in the Server Management window (Figure 21) under **In-use Config Files**. Any changes you made to certain elements in the base server, receiver, or nodes configuration files in use will only be read and followed when you refresh that host. This applies both to changes you make to host configuration files that you uploaded through the Server Management window, and to changes you make by modifying the configuration files directly on the host. The refresh feature in the Server Management window functions the same as the `iwodserverreset` command-line tool. See “Refreshing the OpenDeploy Server” on page 115 for more information, including a list of those supported elements.

For example, if you modify the log file directory of the base server configuration file of a selected host, the new log file directory will only be used after the host is refreshed. Any configuration change you make will only be applied to actions that occur after the host is refreshed. OpenDeploy will not retroactively apply changes, such as moving the older log files from their previous location to the new one you specified.

Refreshing OpenDeploy only applies to the host configuration files whose names appear in the **In-use Config Files** section of the Server Management window. OpenDeploy hosts with the base server software installed will display the base server configuration file name (default `odbase.xml`) and the nodes configuration file (default `odnodes.xml`), while hosts with the receiver software installed will only display the name of the receiver configuration file (default `odrcvr.xml`).

You cannot change the name of host configuration files in use by a selected host through the OpenDeploy user interface, nor can you select a different configuration file. Changing the host configuration files can only be done by modifying the host service configuration file (`deploy.cfg`) and restarting the OpenDeploy services or daemons. See “Modifying the Service Configuration File” on page 65 for more information about modifying the base server service and receiver service configuration files.

As the refresh on the selected host takes place, the progress is logged in the base server or receiver log file. Check these logs to determine when the refresh procedure has completed before resuming OpenDeploy tasks.

This feature is only available to individuals holding an Administrator role on the affected OpenDeploy host. See “Roles and Authorization” on page 132 for more information.



To refresh a selected OpenDeploy host's configuration files, follow these steps:

1. Select **Servers > Manage Servers** to display the Server Management window.
2. Select the name of the OpenDeploy server whose host configuration files you want to refresh from the **Selected Host** list.
3. Click **Refresh Server** to begin the refreshing procedure.
4. Click **View Log** to display a separate browser window containing the OpenDeploy Log Viewer window. Here you can view the base server or receiver log file to follow the progress of the refresh.

## Determining the OpenDeploy Server Version

You can determine which version of OpenDeploy you are running by using the `iwodservergetversion` command-line tool.

To determine the version of your OpenDeploy server, follow these steps:

1. Navigate to the following directory:

```
od-home/bin
```

2. Display the version by entering the following command at the prompt:

```
iwodservergetversion
```

The `-h` option associated with the `iwodservergetversion` command-line tool will display help information.

```
iwodservergetversion -h
```

```
-h
```

Displays help information.

## Displaying the OpenDeploy Server Status

You can display the status of the OpenDeploy server, including its registry port and the number of active deployments, by using the `iwodserverstatus` command-line tool.

To display the status of your OpenDeploy server, follow these steps:

1. Navigate to the following directory:

```
od-home/bin
```

2. Display the status by entering the following command at the prompt:

```
iwodserverstatus
```

There are various options associated with the `iwodserverstatus` command-line tool. Here is a listing of these options, along with the usage syntax:

```
iwodserverstatus [-h | -v | -q]
```

-h

Displays help information.

-v

Displays version information.

-q

Omits displaying the number of active deployments.

## Roles and Authorization

OpenDeploy recognizes two levels, or *roles*, of access to the product's features and functionality:

- Administrator
- User

The role an individual has determines what tasks that person can perform within the OpenDeploy environment, and on specific OpenDeploy hosts.

### Administrator Role

The *Administrator role* allows full access to OpenDeploy configuration and functionality. The Administrator role is authorized to perform the following tasks:

- Scheduling, starting, and cancelling all deployments.
- Monitoring status of all deployments.
- Viewing all schedules for deployments.
- Viewing the XML source code of a deployment configuration.
- Importing a deployment configuration file into OpenDeploy.
- View the OpenDeploy server log.
- View and upload OpenDeploy server configuration files.
- Create deployment configurations.
- Adding additional individuals to Administrator and User roles.
- Designating which individuals holding User roles can invoke particular deployments.

## User Role

The *User role* authorizes an individual with User access to perform deployment operations on specific deployments (previously created by an individual with an Administrator role). Specifically, the User role can perform the following tasks for their assigned deployments:

- Scheduling, starting, and cancelling deployments.
- View the schedules for the deployments.
- Monitoring status.
- Viewing the XML source code of a deployment configuration.
- View the OpenDeploy server log.

For example, both *User1* and *User2* belong to the same Web Operation user group with access to Web server *mars* at the operating system level. *User1* is authorized to manage the deployment for the Residential Mortgage Web site, while *User2* is authorized to manage the deployments for the Brokerage Web site. Both *User1* and *User2* have access to the same development server, but each is allowed to launch only the deployment assigned to them.

## Managing Role Access

Individuals with an Administrator role can assign or revoke Administrator and User roles to others. Role management includes performing the following tasks for each individual:

- Specifying the system role access, which modifies the operations server role database. This determines what role an individual can hold within the OpenDeploy environment as a whole.
- Specifying the server role access, which modifies the OpenDeploy server role database of the particular host server. This determines what role an individual can hold on a specific OpenDeploy host.

The system role access must be specified for an individual before you can specify their server role access.

## Specifying System Role Access

Specifying the system role access of an individual determines under what role that person can log in to the OpenDeploy user interface. The operations server or OpenAPI software maintains the role database for system roles. The role under which the individual logs in to the OpenDeploy environment determines in part what subsequent role that individual can have on a server-by-server basis. Here is a description of each system access role:

- **Admin** — logging in as `admin` gives the following access:
  - Access to all hosts in the OpenDeploy environment.
  - Ability to perform administrator tasks on the OpenDeploy hosts where the individual has been assigned the Administrator role.
- **User** — logging in as `user` gives the following access:
  - Read-only access to all hosts in the OpenDeploy environment.
  - Ability to perform user tasks on the OpenDeploy hosts where the individual has been assigned the User role.

An individual must be assigned a system role prior to being assigned an Administrator or User role on a specific OpenDeploy host.

You can assign and manage system role access through the user interface in the System Roles window (Figure 22).

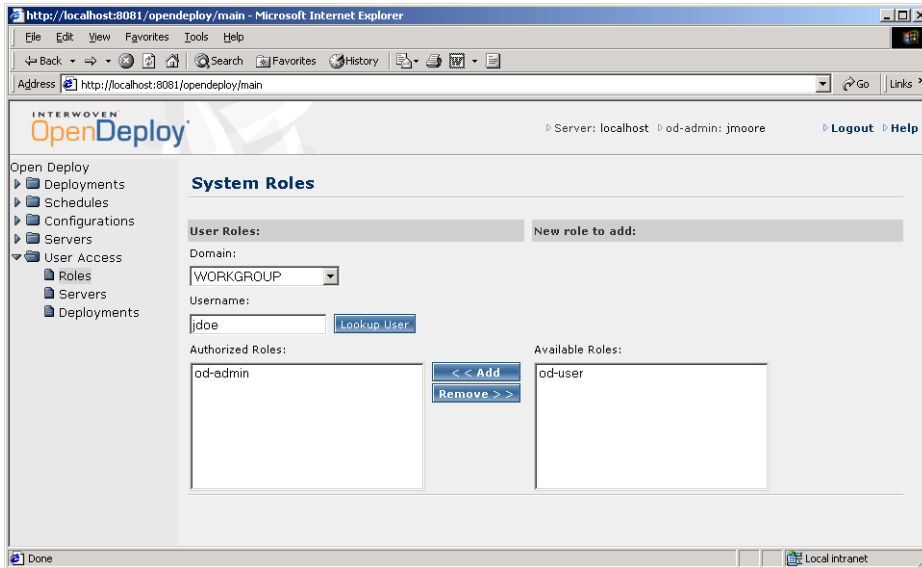


Figure 22: System Roles Window on a Windows Host

The System Roles window allows you to specify an individual in your enterprise and assign that person a system role. All individuals must have an existing user account on the operations server host to be assigned a system role.

Individuals new to the OpenDeploy system role environment will initially have no authorized role established. Individuals who already have a system role assigned will have that role displayed in the **Authorized Roles** list. As the administrator, you must determine the needs of individuals you add, and assign them the appropriate role. If the individual needs administrative (od-admin) access on even one OpenDeploy host, then that individual must be given administrator login access. Otherwise, you give the individual user access (od-user).

To assign or revoke system roles, follow these steps:

1. Select **User Access > Roles** to display the System Roles window.
2. (Windows only) Select the domain containing the individual's system account from the **Domain** list.
3. Enter the individual's user name in the **Username** box and click **Lookup User**. The user name you enter must be valid user account on the operation server host.

If the individual is new to the OpenDeploy environment (does not yet have any roles assigned on the operations server), then both the `od-admin` and `od-roles` options will be listed in the **Available Roles** list. If the individual already has a system role assigned, that role will be displayed in the **Authorized Roles** list.

4. Select the role you want to assign the individual, and click **Add** to move that role to the **Authorized Roles** list.
5. Select any role currently held by the individual from the **Authorized Roles** list, and click **Remove** to move that role to the **Available Roles** list. That individual will no longer have that role access to the OpenDeploy environment.
6. Repeat this procedure for every individual to whom you want to assign or revoke a system role.

After you have established all your system roles, you can assign server roles to each individual.

### Specifying Server Role Access

Specifying the server role access of an individual determines what level of access that individual will have on a particular OpenDeploy host server. The type of role available to assign an individual is determined by what level of system role the individual was granted. An individual assigned with a system role of User (`od-user`) can only be assigned a User role (`od-user`) on a particular host. An individual assigned a system role of Administration (`od-admin`) can be assigned an Administrator role (`od-admin`) on a particular host. An individual with an Administrator or User system role, but no server role, assigned for a particular server can monitor deployments on that OpenDeploy server. However, the individual cannot perform tasks that require an Administrator or User role's authentication.

You can assign and manage server role access through the user interface in the Server Access window (Figure 23).

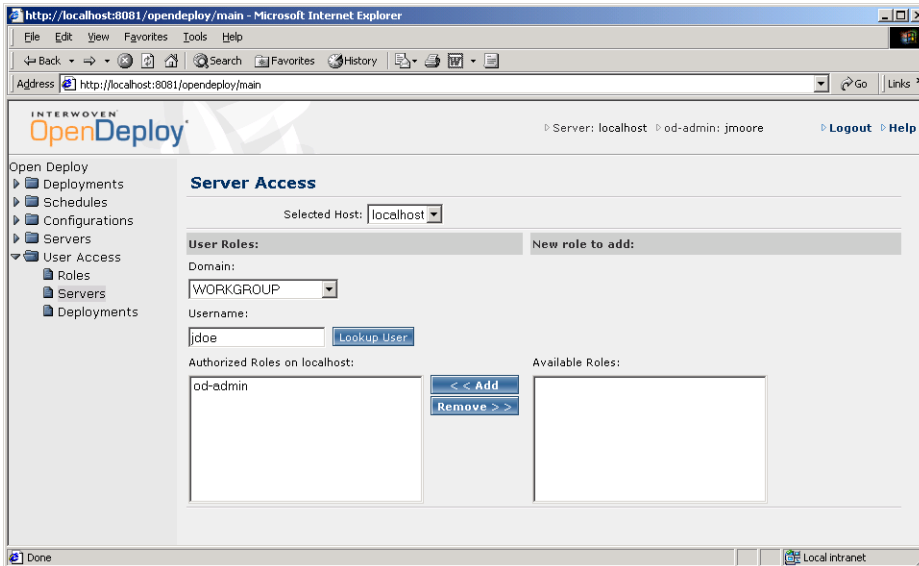


Figure 23: Server Access Window

Assigning or revoking server role access is similar to assigning system role access, except it is done on a per-server basis. You must assign and revoke roles for each server separately.

To assign or revoke server roles, follow these steps:

1. Select **User Access > Servers** to display the Server Access window.
2. Select the OpenDeploy host to which you are assigning roles from the **Selected Host** list.
3. (Windows only) Select the domain containing the individual's system account from the **Domain** list.

4. Enter the individual's user name in the **Username** box and click **Lookup User**.

If the individual is new to that particular OpenDeploy host (does not yet have any roles assigned), then the available role options are listed in the **Available Roles** list, depending on what system roles the individual is assigned:

- Administrator — `od-admin` is listed.
- User — `od-user` is listed.

If the individual already has a role assigned on that OpenDeploy host, that role will be displayed in the **Authorized Roles** list.

5. Select any role you want to assign the individual from the **Available Roles** list, and click **Add** to move that role to the **Authorized Roles** list.
6. Select any role currently held by the individual from the **Authorized Roles** list, and click **Remove** to move that role to the **Available Roles** list. That individual will no longer have that role access to the host server.
7. Repeat this procedure for every individual to whom you want to assign or revoke a server role.

## Assigning User Roles Deployments

Individuals with administrator access to an OpenDeploy host have unlimited access to that host's deployments. An administrator can limit the access of an individual with a User role on that host to only those deployments the administrator assigns. An individual with User role access can perform the following tasks on that host:

- Start and cancel deployments for which the individual is authorized.
- View the XML-based configuration for which the individual is authorized.
- Create and edit schedules for which the individual is authorized.
- Monitor the status of all deployments.
- View the schedules for all deployments.

The same deployment can be assigned to more than one individual with User role access on that host, and those individuals can have any number of deployments assigned them.

You can authorize role access to specific deployments on an OpenDeploy host through the user interface in the Deployment Authorization window (Figure 24).

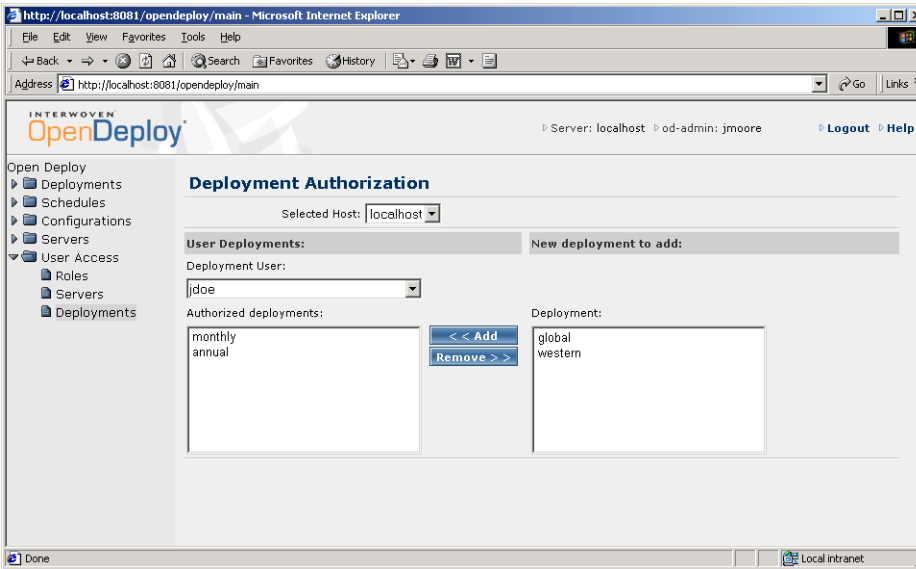


Figure 24: Deployment Authorization Window

To authorize individuals with User roles to perform specified deployments on a particular OpenDeploy host, follow these steps:

1. Select **User Access > Deployments** to display the Deployment Authorization window.
2. Select the OpenDeploy host server whose User roles you want to modify from the **Selected Host** list.
3. Select the user for whom you are authorizing deployments from the **Deployment User** list.

This list will display those individuals who have User or Administrator role access to the OpenDeploy host you selected. Those deployments for which the individual already has authorization to use are displayed in the **Authorized Deployments** list. Those deployments available to be assigned are displayed in the **Deployment** list.

4. Select any deployment you want to disallow the user to control from the **Authorized Deployments** list and click **Remove**. Those deployments are now displayed in the **Deployment** list.
5. Select any deployment you want to authorize the user to perform from the **Deployment** list and click **Add Deployment**. That deployment now appears in the **Authorized Deployments** list.

## Composing Deployments

You can create a new deployment configuration file, or edit an existing one, using the following methods:

- Using a text or XML Editor
- Using the Deployment Configuration Composer

### Using a Text or XML Editor

Because deployment configuration files are XML-based, you can use your favorite text or XML editor to open and modify any of them. Using a text or XML editor to edit configuration files requires you to have file system access to the OpenDeploy host where the deployment you want to create or modify resides. All new and modified deployment configuration files must reside in the following location:

*od-home/conf*

for OpenDeploy to use them.

Modifying deployment configuration files requires an understanding of XML syntax, and of the deployment configuration DTD. You should not try to modify deployment configuration file unless you have expertise in both. See Chapter 4, “Deployment Configurations” for more information on modifying deployment configuration files.

## Using the Deployment Configuration Composer

The Deployment Configuration Composer is a tool in the OpenDeploy user interface that allows you to create and modify existing deployment configurations without having to edit the files using a text or XML editor. Knowledge of XML is not required to use this tool. However, you do need to understand the OpenDeploy features and functionality described in Chapter 4, “Deployment Configurations” and Chapter 5, “Advanced Features” before you can create a complete deployment configuration. See Chapter 6, “Composing Deployments” for more information on how to use this tool.

## Running Deployments

This section describes how start, cancel, and monitor deployment configurations.

An individual holding an Administrator role on the OpenDeploy server can start and cancel any deployment on that host. Individuals holding User roles on that host only can start and cancel deployments on that host that are assigned to them. Individuals holding either type of role can view the progress and status of any deployment.

### Starting a Deployment

You should perform a test deployment using the `test.xml` deployment configuration before trying any other deployments. Performing the test will ensure that your OpenDeploy server is properly configured, and will give you practice with deployments. See “Performing a Test Deployment” on page 145 for more information.

You can start a deployment through the OpenDeploy user interface (Figure 25).

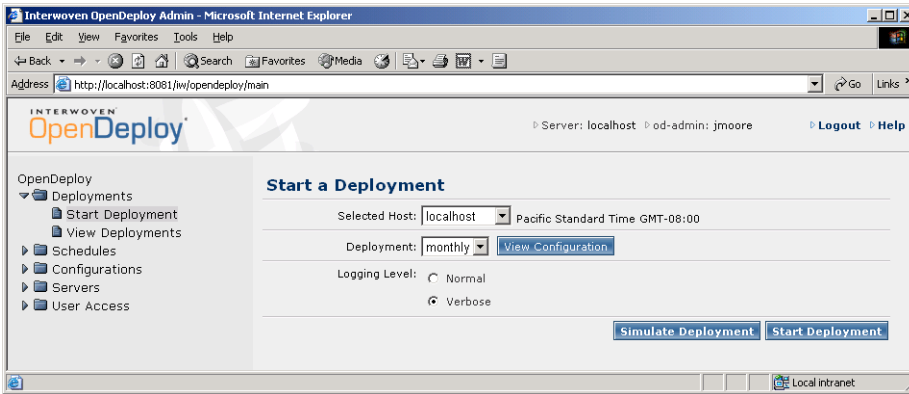


Figure 25: Start a Deployment Window

You can also start a deployment by entering the appropriate command-line tool at the command prompt.

## User Interface

To start a deployment using the OpenDeploy user interface, follow these steps:

1. Select **Deployments > Start Deployment** to display the Start Deployment window.
2. Select the OpenDeploy server you want to act as the source of the deployment from the **Selected Host** list. When you select a particular host, its deployment choices become available in the **Deployment** list.
3. Select the deployment you want to start from the **Deployment** list. If you are performing an initial test of OpenDeploy, select **test**.
4. Select your choice from one of the following **Logging Level** options:
  - **Verbose** — logs high level of detail on deployment events as they occur. This logging level is best suited for troubleshooting deployment problems or evaluating deployment performance. Verbose logging can create very large log files. This is the default logging level.
  - **Normal** — logs standard status and error messages. In most cases, this level of logging provides a sufficient amount of detail to meet your needs.

5. Click **Simulate Deployment** if you want to perform a simulated deployment before actually moving files to the target hosts. Otherwise, go on the next step. See “Performing a Simulated Deployment” on page 145 for more information.
6. Click **Start Deployment** to start the deployment. The Deployment Started window appears (Figure 26), displaying information regarding the deployment you just started.

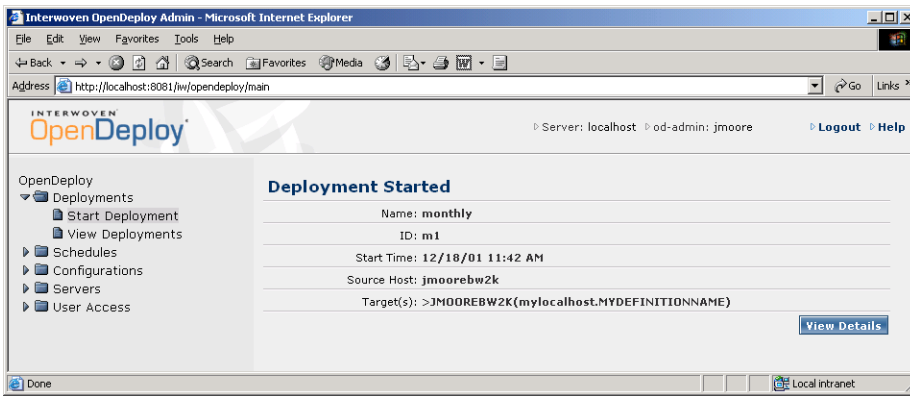


Figure 26: Deployment Started Window

## Command Line

Command-line tools only can be issued on the host where the OpenDeploy server is installed. Commands can be issued by anyone regardless of whether they hold an Administrator or User role. There are no authentication or authorization checks on individuals issuing these commands.

To start a deployment using the command line, follow these steps:

1. Navigate to the following directory:
2. Start the deployment by entering the following command at the prompt:

```
od-home/bin
```

```
iwodstart deployment
```

where *deployment* is the name of the deployment you want to start.

There are various options associated with the `iwodstart` command-line tool. Here is a listing of these options, along with the usage syntax:

```
iwodstart -h | -v
```

```
iwodstart deployment [-async] [-inst instance] [-k "key=value"]+ [-sim]  
[-V (normal | verbose)]
```

<code>-h</code>	Displays usage information.
<code>-v</code>	Displays version information.
<code><i>deployment</i></code>	Name of the deployment to start.
<code>-async</code>	Runs <code>iwodstart</code> command asynchronously. The <code>iwodstart</code> command will return before the deployment completes.
<code>-inst</code>	Includes the deployment instance name <i>instance</i> , which is a suffix that is appended to the deployment name. This option is used to create unique deployment names for each instance of a deployment configuration.
<code>-k <i>arg</i></code>	Key/value substitution with " <i>key=value</i> " as the <i>arg</i> value.
<code>-sim</code>	Enables the simulated deployment feature.
<code>-V <i>arg</i></code>	Logging level with <code>verbose</code> or <code>normal</code> as args.

The `iwodstart` command returns the following codes regarding the status of the deployment:

- 0 - succeeded
- 1 - starting of the deployment failed
- 2 - deployment ran and returned a failed status

## Performing a Test Deployment

After you have installed and configured OpenDeploy, you should perform a test deployment to ensure everything is working correctly. The OpenDeploy software includes a test deployment configuration `test.xml` that will deploy files from one location to another on your OpenDeploy host. The test configuration only uses default settings present in the host configurations files, so no further configuration is required on your part.

The test deployment configuration will move the following file:

```
od-home/conf/dtd/oddeployment.dtd
```

to the following location:

```
od-home/tmp
```

Use either the OpenDeploy user interface or command-line method to start the deployment. See “Starting a Deployment” on page 141 for more information. If you successfully deployed the file to its designated target location on your server, then you are ready to perform a deployment to a target on another host.

## Performing a Simulated Deployment

You can perform a simulated deployment of any deployment configuration using the *Simulate Deployment* feature. The Simulate Deployment feature performs a simulated deployment that does not actually transfer any content over to the target, but logs what would have been transferred over. Using this feature allows you to test out and see what would have been transferred if the deployment was actual. The record of this result is in the deployment log files.

To perform a simulated deployment, follow these steps:

1. Select **Deployments > Start Deployment** to display the Start Deployment window.
2. Select the OpenDeploy server you want to host the simulated deployment from the **Selected Host** list. When you select a particular host, its deployment choices become available in the **Deployment** list.
3. Select the deployment you want to simulate from the **Deployment** list.

4. Select your choice from one of the **Logging Level** options. This is the level of logging that will be performed for your simulated deployment. See “Starting a Deployment” on page 141 for more information.
5. Click **Simulate Deployment** to begin the simulation. The Deployment Started window appears.
6. Click **View Details** to display the View Deployments window. Here you can view deployment information for your simulated deployment just like an actual deployment.
7. Click **View Log** to view logging information. Here you can view the list of files that would have been included in an actual deployment.

After evaluating the simulated deployment, you can actually deploy the files to the target hosts by clicking **Start Deployment**.

## Checking File Integrity on Production Servers

The Simulate Deployment feature described in the previous section can also serve as a valuable tool in ensuring the integrity of Web site files residing on your production servers. You can use this feature to determine if a targeted production server is out of sync with your development server. Running the Simulate Deployment feature will create an entry for any file that would be deployed in the deployment log file. A file that was deployed unexpectedly is indicative of a file being mistakenly or intentionally added, deleted, or changed. Use the Simulate Deployment feature regularly to ensure the integrity of your production server Web sites. See “Performing a Simulated Deployment” on page 145 for information on using the Simulate Deployment feature.

## Viewing the Source Code of a Deployment Configuration

You can view the XML-based source code of a selected deployment configuration's file (Figure 27) within the OpenDeploy user interface.

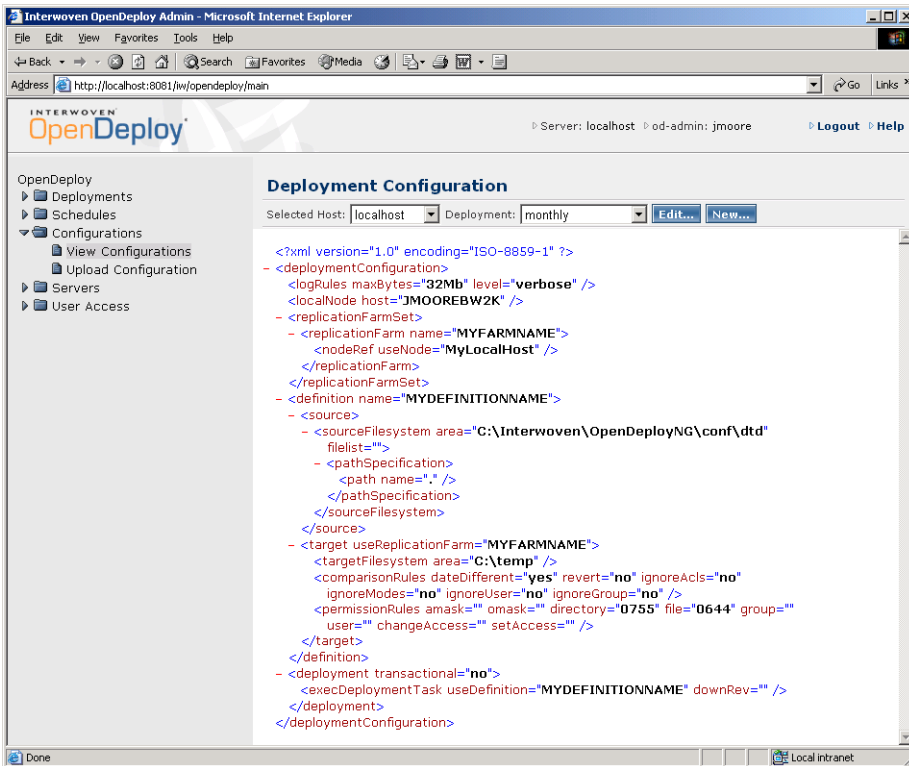


Figure 27: View Configuration Window

Viewing the source code of a deployment configuration allows you to identify and troubleshoot a deployment configuration file issue quickly. You can also use this feature simply to see what element and attribute values are present. However, you cannot actually edit a deployment configuration displayed in the Deployment Configuration window. Instead you must edit the deployment configuration either using a text or XML editor, or using the Deployment Configuration Composer. See “Composing Deployments” on page 140 for more information.

To view the source code of a deployment configuration, follow these steps:

1. Select **Configurations > View Configuration** to display the View Configuration window.
2. Select the name of the OpenDeploy server hosting the deployments whose configuration information you want to view from the **Selected Host** list.
3. Select the name of the deployment whose configuration information you want to view from the **Deployment** list. This information is displayed in the window below.

You can also view the configuration of a selected deployment in the Start Deployment window by clicking **View Config**.

## Monitoring Deployments

You can view information regarding the deployments being sent in the Source Deployments window and received in the Target Deployments window. These windows include information on deployments that have already taken place, that are currently in progress, and that are pending. You can also access log information and other details on a specific deployment.

To monitor the progress of your deployments, follow these steps:

1. Select **Deployments > View Deployments** to display the Source Deployments window.
2. Select the OpenDeploy server whose deployments you want to monitor from the **Selected Host** list.
3. Select one of the following choices from the **View** list:
  - **Sending** — select to display the Source Deployments window. Here information on deployments initiated by the host server is displayed. See the following section for details on the contents of the Source Deployments window.
  - **Receiving** — select to display the Target Deployments windows. Here information on deployments received by the host server is displayed. See the following section for details on the contents of the Target Deployments window.

## Source Deployments Window

The Source Deployments window (Figure 28) appears when you select **Sending** from the **View** list. The Source Deployments window contains information regarding deployments originating from the selected OpenDeploy host. In this example, the OpenDeploy host has sent the deployment **yearly (16)**. By selecting the **yearly (16)** link in the upper table, the **Details** table appears below it, displaying information specific to the yearly (16) deployment.

The screenshot shows the Interwoven OpenDeploy Admin interface in a Microsoft Internet Explorer browser window. The address bar shows `http://localhost:8081/lw/opendeploy/main`. The page title is "Interwoven OpenDeploy Admin - Microsoft Internet Explorer".

On the left is a navigation sidebar with the following items:

- OpenDeploy
  - Deployments
    - Start Deployment
    - View Deployments
  - Schedules
  - Configurations
  - Servers
  - User Access

The main content area is titled "Source Deployments". It includes a "Selected Host" dropdown set to "localhost" and a time zone selector set to "Pacific Standard Time GMT-08:00". Below this is a "View:" dropdown set to "Sending", and checkboxes for "active", "completed", and "pending", along with a "1 day(s)" filter and an "Update" button.

The main table lists deployments:

Name (ID)	Start	Target	Status	Elapsed	Owner	Actions
<a href="#">monthly (m3)</a>	12/18/01 12:08 PM	JMOOREBW2K(mylocalhost.MYDEFINITIONNAME)	running	0:00:08	interwoven\jmoore	<a href="#">View Log...</a>
<a href="#">test (m2)</a>	12/18/01 12:00 PM	JMOOREBW2K(mylocalhost.MYDEFINITIONNAME)	completed	0:01:58	interwoven\jmoore	<a href="#">View Log...</a>
<a href="#">monthly (m1)</a>	12/18/01 11:42 AM	JMOOREBW2K(mylocalhost.MYDEFINITIONNAME)	failed	0:00:00	interwoven\jmoore	<a href="#">View Log...</a>

Below the table is a section titled "monthly (m3) - Details" with a "Refresh" button. It contains a table with deployment details:

Target Host	Progress	Elapsed	Average Data Rate	Type	Actions
JMOOREBW2K(mylocalhost.MYDEFINITIONNAME)	setup transferring	0:00:11	1 KB/s	directory comparison	<a href="#">View Log...</a>

Figure 28: Source Deployments Window

If you display the Source Deployments window while the deployment is in progress, the **Details** table will indicate the deployment is still in progress, and under certain circumstances gives you the option of cancelling the deployment. See “Cancelling Deployments in Progress” on page 153 for more information.

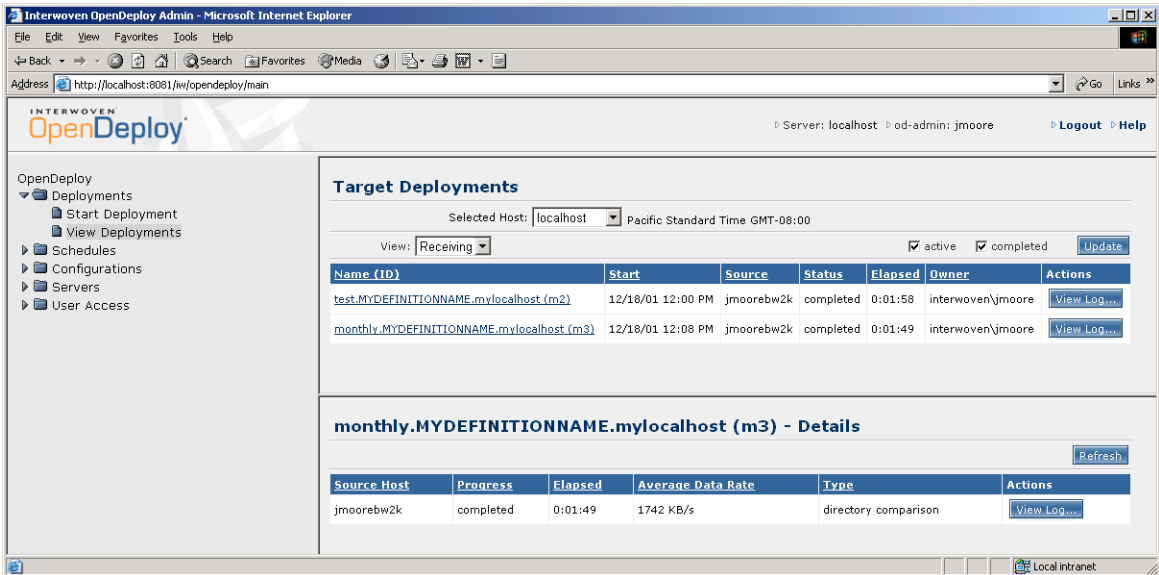
If you click **Refresh** after the deployment has completed, the **Details** table will change to reflect the deployment in its finished state (Figure 29):

monthly (m3) - Details					
<a href="#">Refresh</a>					
Target Host	Progress	Elapsed	Average Data Rate	Type	Actions
JMOOREBW2K(mylocalhost.MYDEFINITIONNAME)	completed	0:01:49	1742 KB/s	directory comparison	<a href="#">View Log...</a>

Figure 29: Source Deployments Window — Details Table

## Target Deployments Window

The Target Deployments window (Figure 30) appears when you select **Receiving** from the **View** list.



The screenshot shows the Interwoven OpenDeploy Admin interface in a Microsoft Internet Explorer browser window. The address bar shows the URL `http://localhost:8081/lw/opendeploy/main`. The page title is "Interwoven OpenDeploy Admin - Microsoft Internet Explorer".

The main content area is titled "Target Deployments". It includes a "Selected Host" dropdown set to "localhost" and a "Pacific Standard Time GMT-08:00" time zone. Below this is a "View" dropdown set to "Receiving". There are checkboxes for "active" and "completed", and an "Update" button.

Name (ID)	Start	Source	Status	Elapsed	Owner	Actions
test.MYDEFINITIONNAME.mylocalhost (m2)	12/18/01 12:00 PM	jmoorebw2k	completed	0:01:58	interwoven\jmoore	<a href="#">View Log...</a>
monthly.MYDEFINITIONNAME.mylocalhost (m3)	12/18/01 12:08 PM	jmoorebw2k	completed	0:01:49	interwoven\jmoore	<a href="#">View Log...</a>

Below the table is a section titled "monthly.MYDEFINITIONNAME.mylocalhost (m3) - Details". It includes a "Refresh" button and a table with the following data:

Source Host	Progress	Elapsed	Average Data Rate	Type	Actions
jmoorebw2k	completed	0:01:49	1742 KB/s	directory comparison	<a href="#">View Log...</a>

Figure 30: Target Deployments Window

The Target Deployment window contains information regarding deployments being received by the selected OpenDeploy host. In this example, the OpenDeploy host has received the deployment `monthly.MYDEFINITIONNAME.mylocalhost (m3)`. By selecting the **monthly.MYDEFINITIONNAME.mylocalhost (m3)** link in the upper table of the Target Deployments window, the **Details** table appears below it, displaying information for that deployment.

## Viewing Options

The viewing options are similar for both the Source Deployments and Target Deployments windows:

- **Active** check box — check to view deployments that are currently active.
- **Completed** check box — check to view deployments that have been completed.
- **Pending** check box (Source Deployments window only) — check to view scheduled deployments that have not occurred yet. You can specify the deployments pending to a specified number of days in the future by entering that number in the text box.

Click **Update** after changing the viewing options to refresh the window.

## Deployments Table

The **Deployments** table displays the following information regarding all deployments being sent or received by the selected host, depending on whether the Source and Target Deployments window is being displayed.

- **Name (ID)** column — displays the name and identification number of the deployment. Selecting a deployment name displays its details in the **Details** table. On the receiver, the name value is:

*deployment.definition.target-server*

where the following variables apply:

- *deployment* — the name of the associated deployment.
- *definition* — the name of the definition in the deployment configuration that contains the source/target pairing.
- *target-server* — the logical name of the target host server receiving a deployment as it appears in the nodes configuration file of the sending host.
- **Start** column — displays the date and time the deployment started.
- **Target** column (Source Deployments window only) — displays the name of each target host receiving the deployment.
- **Source** column (Target Deployments screen only) — displays the name of the source host sending the deployment.

- **Status** column — displays the current status of the deployment, such as whether it is running, pending, completed, or failed.
- **Elapsed** column — displays the elapsed time the deployment has been running.
- **Owner** column — displays the login name of the deployment's owner.

### Details Table

The **Details** table displays information regarding the source host or target hosts participating in the deployment you selected. If the Source Deployments window is displayed and you select a deployment with multiple target hosts, each of those target hosts will be displayed in the **Details** table.

The **Details** table displays the following information regarding specific deployments sent or received by the host:

- **Target Host** (Source Deployments window only) — displays the name of the server receiving the deployment as it appears in the nodes configuration file of the sending host.
- **Source Host** (Target Deployments window only) — displays the name of the server sending the deployment.
- **Progress** column — displays the status and particular activity taking place regarding the deployment at that given time.
- **Elapsed** column — displays the elapsed time the deployment has been running.
- **Average Data Rate** column — displays the average data transmission rate of the deployment at that given time.
- **Type** column— displays the type of deployment, such as directory comparison, TeamSite comparison, file list, and others.

Click **Refresh** to update the **Details** table with the latest information on the selected deployment, such as whether a deployment in progress has completed yet.

## Deployment Logging

Clicking **View Log** for either a deployment listed in the Source Deployment window, or one of the deployment's corresponding source or target host listings in the **Details** table will display various types of logging information. See “Logging” on page 155 for more information.

## Cancelling Deployments in Progress

You can cancel a deployment in progress during certain stages of the deployment process depending on the type of deployment:

- Initial setup — all deployments
- Compare phase — deployments that compare files only
- Pre-commit phase — transactional deployments only

If you choose to cancel a deployment during one of these phases, OpenDeploy will complete the phase in progress and then cancel the remainder of the deployment. In some cases, a lengthy amount of time can pass between the time you order the cancellation and when OpenDeploy completes the phase in progress before quitting the deployment. You cannot cancel a deployment once the pre-commit phase is over.

### Initial Setup

All deployments take time to set up the deployment before files are actually compared or moved. A larger deployment with more target hosts take longer to perform its initial setup than a smaller deployment with a lower number of targets. Any deployment can be canceled during its setup phase.

### Compare Phase

The *compare phase* is when OpenDeploy compares the files between file system locations or TeamSite areas. Those deployments that compare files can be canceled during their compare phases as well as their initial setup. The length of the compare phase is dependent on the number of files being compared. A small number of files in a deployment, even if their total file size is large, will result in a brief compare phase. A large number of files, even if the total file size is smaller, will result in a longer compare phase.

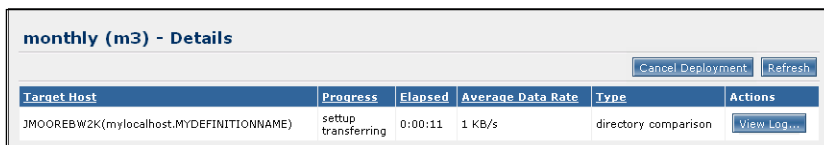
Deployment types that do not compare files, such as file list deployments, do not have a compare phase. These can only be canceled during their initial setup.

## Pre-Commit Phase

Transactional deployments can be canceled before or during their *pre-commit phase* in addition to their initial setup and compare phases. The pre-commit phase is when OpenDeploy determines whether or not all the targets have successfully received their deployments.

## Deployment Cancellation Window

The time between the start of the deployment and the end of the initial setup, compare, and pre-commit phases is known as the *deployment cancellation window*. If the cancellation window is open, the **Cancel Deployment** button will be displayed in the **Details** table of the Source Deployments window (Figure 31). Clicking **Cancel Deployment** stops the deployment at that point. In some cases, the deployment cancellation window is too short to permit cancellation of the deployment.



monthly (m3) - Details					
Target Host	Progress	Elapsed	Average Data Rate	Type	Actions
JMOOREBWZK(mylocalhost.MYDEFINITIONNAME)	setup transferring	0:00:11	1 KB/s	directory comparison	<a href="#">View Log...</a> <a href="#">Cancel Deployment</a> <a href="#">Refresh</a>

Figure 31: Details Table with Cancel Deployment Button

You can only cancel a deployment in progress from the source host using the OpenDeploy user interface. A target host cannot cancel a deployment it is receiving. There is no method for cancelling a deployment from the command prompt.

You cannot restart a deployment after it has been cancelled. If a transactional deployment has been cancelled, the deployment is considered to have failed and is rolled back with the targets restored to their previous states.

Depending on the speed of the deployment phases and when you issue the cancellation order, it is possible that a deployment you attempt to cancel will be completed anyway. The deployment cancellation window might close before OpenDeploy can process the deployment cancellation order after you click **Cancel Deployment**. In other cases, the cancellation window can close before the user interface can fully display the Details window with the **Cancel Deployment** button displayed.

To cancel a deployment in progress sent by your host, follow these steps:

1. Select **Deployments > View Deployments** after a deployment has started to display the Source Deployments window.

If you started the deployment manually, you can click **View Details** in the Deployment Started window to display the Source Deployments window and the **Details** table for your deployment. Skip to step 4.

2. Select the OpenDeploy server whose deployment you want to cancel from the **Selected Host** list.
3. Select the link of the deployment you want to cancel. That deployment's target hosts are displayed in the **Details** table.

If the cancellation window for the deployment is still open, the **Cancel Deployment** button is displayed for that target.

4. Click **Cancel Deployment** to stop the deployment for each target host you want.

## Logging

OpenDeploy provides a variety of different types of logging information including:

- Activities involving the base server or receiver host (base server or receiver log)
- Activities involving a deployment as a whole (macro deployment log) from both the sending and receiving hosts
- Activities involving a specific source/target pair within a deployment (micro deployment log) from both the sending and receiving hosts

You can view and analyze logging information to determine the efficiency of your deployments, whether they are successful or not, and for general troubleshooting.

## Log File Location

The default location for all log files is:

*od-home/log*

You can specify another location for the base server and receiver host log files by entering the path in the `directory` attribute of the `logRules` element in the corresponding base server or receiver configuration file (by default `odbase.xml` or `odrctr.xml`). However, you cannot specify a different log file directory location in a deployment configuration. See “Logging Configuration Settings” on page 171 for more information.

## Viewing Log Information

You can view log files using one of the following methods:

- Text editor
- OpenDeploy user interface

### Viewing Log Files from a Text Editor

OpenDeploy log files are standard text files that can be opened with any standard text editor, including `vi` and Notepad.

## Viewing Log Files from the OpenDeploy User Interface

You can view log files in the OpenDeploy user interface using the OpenDeploy Log Viewer window (Figure 32). The OpenDeploy Log Viewer window is a separate browser window that appears when you click a **View Log** button in a window.

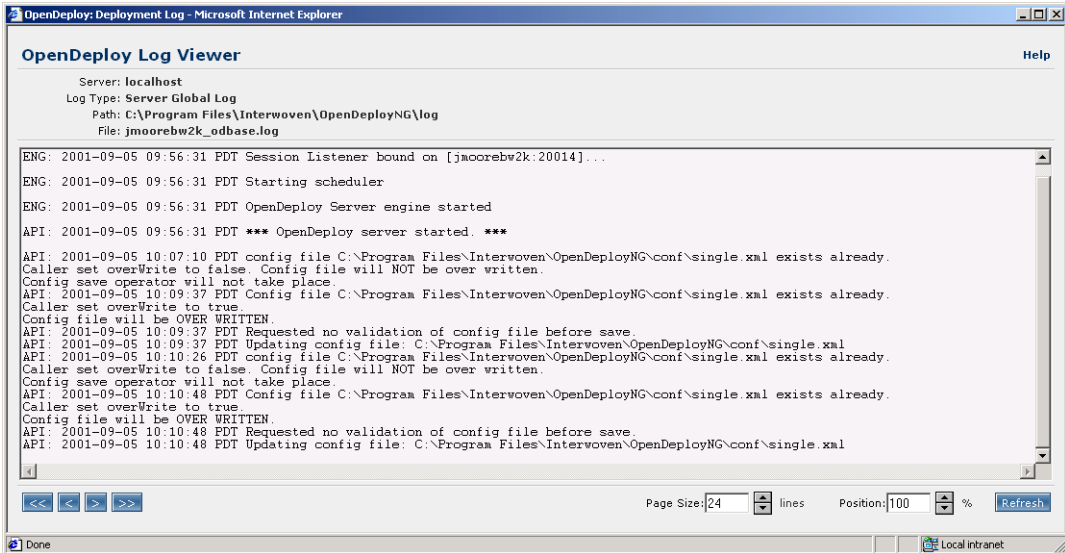


Figure 32: OpenDeploy Log Viewer Window

Each log you select to view is displayed in a separate browser window which allows you to view multiple logs simultaneously.

The format and structure of the various logs are essentially the same. The deployment log windows include the name of the deployment associated with the logs. Here is a description of the log windows:

- **Server** — displays the name of the OpenDeploy host sending and receiving the deployment.
- **Log Type** — displays the type of log file being displayed, such as a server global log (base server or receiver host log) or a deployment macro or micro log for a deployment sent or received.
- **Deployment** (deployment logs only) — displays the name of the deployment associated with the displayed log.

- **Path** — displays the absolute path to the directory containing the log file being displayed.
- **File** — displays the name of the log file being displayed. The following types of log files can be displayed in this window:
  - *server\_odbase.log* — indicates the log file is a base server log.
  - *server\_odrcvr.log* — indicates the log file is a receiver log.
  - *src.deployment.log* — indicates the log file is a source macro deployment log.
  - *rcv.deployment.definition.target-server.log* — indicates the log file is a receiver macro deployment log.
  - *src.deployment.definition.source-server.to.target-server.log* — indicates the log file is a source micro deployment log.
  - *rcv.deployment.definition.source-server.to.target-server.log* — indicates the log file is a receiver micro deployment log.

where the following variables apply:

- *deployment* — the name of the associated deployment.
- *definition* — the name of the definition in the deployment configuration that contains the source/target pairing.
- *source-server* — the name of the source host sending the deployment.
- *target-server* — the logical name of the target host server receiving a deployment as it appears in the nodes configuration file of the sending host.
- << button — click to display the beginning of the log.
- < button — click to display the previous portion of the log.
- > button — click to display the next portion of the log.
- >> button — click to display the end of the log.
- **Page Size** box — enter the number of lines of the deployment log you want to view. You can enter the exact number, or click the arrow buttons up and down in increments of 10 from the existing number. You can range in size from 10 to 1000 lines. You must click **Refresh** to implement the number you entered.

- **Position** box — enter the proportional location percentage (0-100) of the log file to be displayed. You can enter the exact number, or click the arrow buttons up and down in increments of 10. For example, the beginning of the log would be 0, while the center would be 50. You must click **Refresh** to implement the number you entered.
- **Refresh** button — click to refresh the log and to read in fresh data with the **Page Size** and **Position** values you entered.

## Base Server Logging

All activities concerning the OpenDeploy base server host are written to the *base server log*. Base server log entries include information on:

- Starting up OpenDeploy services and daemons
- Adding, removing, and modifying the Administrator and User roles of individuals
- Starting deployments
- Receiving deployments
- Adding schedules for deployments
- Starting a scheduled deployment
- Requests from individuals with User roles that have been denied due to insufficient authorization
- Error information on requested operations

Reviewing the base server log is an effective method of determining the activities of your OpenDeploy sending host, and of troubleshooting problems.



Here is an example of base server log entries:

```
BEGIN LOG: Logfile [C:\Interwoven\OpenDeployNG\log\jmoorebw2k_odbase.log] -----  
--  
API: 2001-11-12 13:09:55 PST GMT-08:00 Using time zone: Pacific Standard Time  
API: 2001-11-12 13:09:55 PST GMT-08:00 Using locale: en_US  
API: 2001-11-12 13:09:55 PST GMT-08:00 Using OpenDeploy home directory:  
C:\INTERW~1\OPENDE~1  
API: 2001-11-12 13:09:55 PST GMT-08:00 Using server config file specified in  
deploy.cfg: C:\INTERW~1\OPENDE~1\etc\odbase.xml  
API: 2001-11-12 13:09:55 PST GMT-08:00 Using server nodes config file specified in  
deploy.cfg: C:\INTERW~1\OPENDE~1\etc\odnodes.xml  
API: 2001-11-12 13:09:59 PST GMT-08:00 Using server log directory  
C:\Interwoven\OpenDeployNG\log specified in server config file.  
API: 2001-11-12 13:09:59 PST GMT-08:00 Using OpenDeploy Server log file  
C:\Interwoven\OpenDeployNG\log\jmoorebw2k_odbase.log.
```

By default, the base server log file resides in the following location:

`od-home/log/server_odbase.log`

where *server* is the name of the host server. If your OpenDeploy host was named *mars*, then the base server log file path and name would be:

`od-home/log/mars_odbase.log`

To access the base server log from the user interface, follow these steps:

1. Select **Servers > Manage Servers** to display the Manage Servers window.
2. Select the host whose source base server log you want to view from the **Selected Host** list.

3. Click **View Log**. A separate browser window appears displaying the OpenDeploy Log Viewer window containing the base server log (Figure 34).

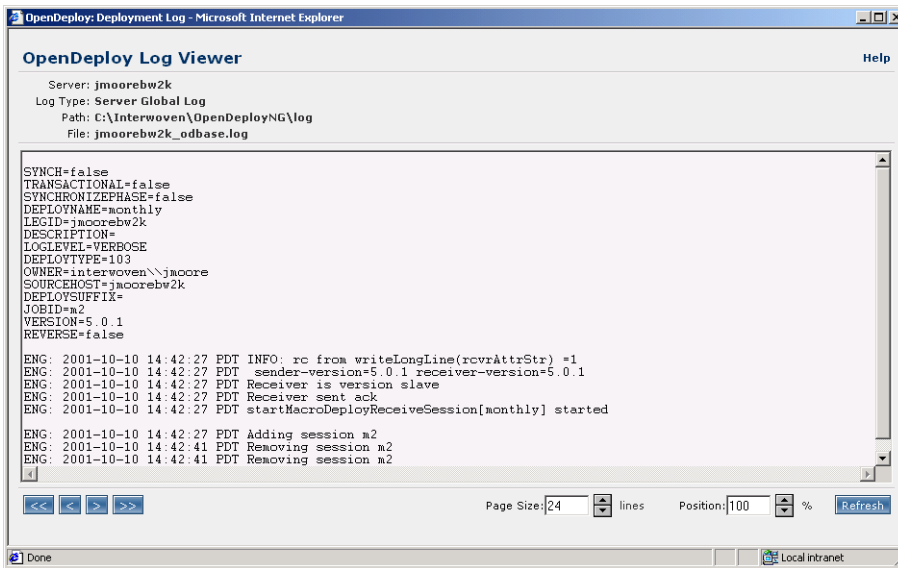


Figure 33: Base Server Log

## Receiver Logging

All activities concerning the OpenDeploy receiver host are written to the *receiver* log. Receiver log entries include information on:

- Starting up OpenDeploy services and daemons
- Receiving deployments

Reviewing the receiver log is an effective method of determining the activities of your OpenDeploy receiving host, and of troubleshooting problems.



By default, the log file resides in the following location:

```
od-home/log/server_odrcvr.log
```

where *server* is the name of the receiver host. If your OpenDeploy host was named *venus*, then the receiver log file path and name would be:

```
od-home/log/venus_odrcvr.log
```

You can view the receiver log from the OpenDeploy user interface in the same manner as for the base server log. See “Base Server Logging” on page 159 for more information.

## Macro Deployment Logging

The *macro deployment* logs write entries on every aspect of a deployment each time it is run. There are two macro deployment logs, one for the source host (the source macro deployment log) and one for the target host (the receiver macro deployment log). If the deployment is configured as a fan-out deployment with multiple target hosts, the macro deployment log will have entries written for each source/target host pairing. Each new running of a deployment causes a new set of log entries to be appended onto the file, so you can review the history of the deployment over a period of time.

Macro deployment log entries include information on:

- Whether or not deployments to each target host were successful.
- Time the deployments took.

Reviewing the macro deployment log is a way to determine how a particular deployment functions, and to troubleshoot problems with that deployment. Here is an example of macro deployment log entries:

```
NG: 2001-11-28 13:06:12 PST GMT-08:00
internalDepName=.monthly.MYDEFINITIONNAME.jmoorebw2k
ENG: 2001-11-28 13:06:12 PST GMT-08:00 Got converted config for
.monthly.MYDEFINITIONNAME.jmoorebw2k
ENG: 2001-11-28 13:06:12 PST GMT-08:00 Waiting for 2 children to complete phases
ENG: 2001-11-28 13:06:12 PST GMT-08:00 All 2 children completed their phases
ENG: 2001-11-28 13:06:12 PST GMT-08:00 Deployment[monthly] Elapsed Time=120 ms
ENG: 2001-11-28 13:06:12 PST GMT-08:00 End logfile
[C:\Interwoven\OpenDeployNG\log\src.monthly.log]
```

### Source Macro Deployment Log

The *source macro deployment log* file contains log entries for a deployment where the OpenDeploy host is the sender.

The source macro log by default resides in the following location:

```
od-home/log/src.deployment.log
```

where *deployment* is the name of the deployment configuration. If your deployment was named *monthly*, then the source macro deployment log file path and name would be:

```
od-home/log/src.monthly.log
```

To access the source macro deployment log from the user interface, follow these steps:

1. Select **Deployment > View Deployment** to display the Source Deployment window.
2. Select the host containing the deployment whose source macro deployment log you want to view from the **Selected Host** list.
3. Select **Sending** from the **View** list. All the deployments that the host sends are displayed in a table.

- Click **View Log** for the deployment whose source macro deployment log you want to view. A separate browser window appears displaying the OpenDeploy Log Viewer window containing the source macro deployment log (Figure 34).

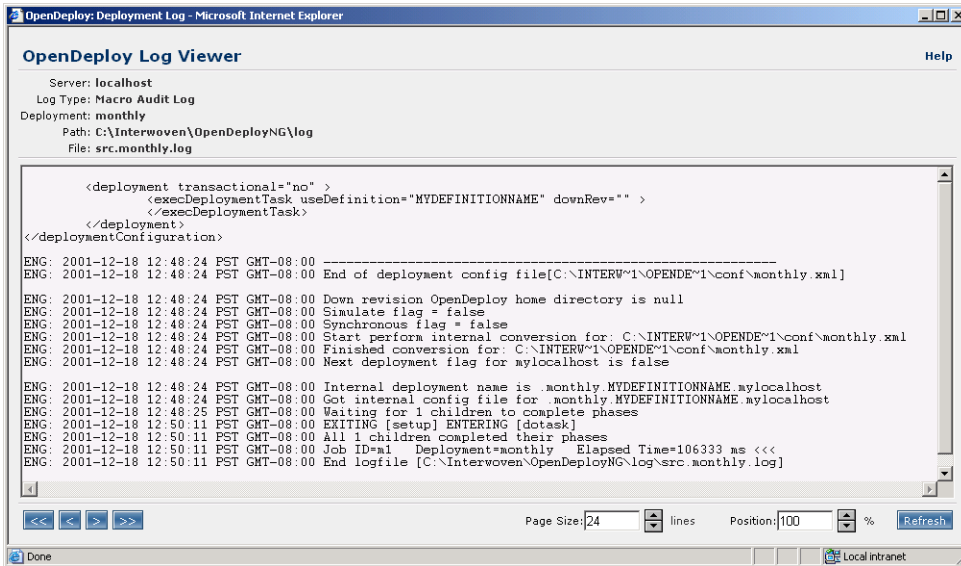


Figure 34: Source Macro Deployment Log

## Receiver Macro Deployment Log

The *receiver macro deployment log* provides a similar service for OpenDeploy hosts receiving deployments as the source macro deployment log does for sending hosts. The receiver macro deployment log contains macro-type entries for the deployments received by the host.

A separate receiver macro log is generated anytime the combination of deployment name, definition name, and logical target name is unique. For example, if a deployment has three separate definitions all pointed at the same target host, three separate receiver macro log files will be generated on the host receiving the deployment.

The receiver macro log by default resides in the following location:

```
od-home/log/rcv.deployment.definition.target-server.log
```

where the following variables apply:

- *deployment* — the name of the associated deployment.
- *definition* — the name of the definition in the deployment configuration that contains the source/target pairing.
- *target-server* — the logical name of the target host server receiving a deployment as it appears in the nodes configuration file of the sending host.

If your deployment was named *monthly* and the definition was named *corporate*, and the target host's logical name is *jupiter*, then the receiver macro deployment log file path and name would be:

```
od-home/log/rcv.monthly.corporate.jupiter.log
```

You must select **Receiving** from the **View** list in the Source Deployments window to access receiver macro deployment logs. See “Source Macro Deployment Log” on page 163 for more information.

## Micro Deployment Logging

The *micro deployment* logs write entries for each source/target host pair in a deployment. If the deployment includes only a single source host and target host, then one micro deployment log each is generated on the source and target hosts. If the deployment is a fan-out type with several target hosts, then a micro deployment log is generated for each of those target hosts.

The source host will generate a separate micro deployment log (the source micro deployment log) for each target host. Each target host receiving the deployment generates its own log (the receiver micro deployment log). Each running of the deployment results in a new set of log entries to be appended onto the file, so you can review the history of the deployment over multiple runnings.

Micro deployment log entries include information on:

- Contact made with the source or target host
- Directories and files successfully deployed
- Directories and files that failed to deploy

Reviewing the micro deployment log is a way to determine how a particular deployment functions, and to troubleshoot problems with that source or target host participating in a deployment. Here is an example of macro deployment log entries:

```
Directories deployed : 2  Files deployed : 34  Links deployed : 0
Directories failed   : 0  Files failed   : 0  Links failed   : 0
Directories deleted  : 0  Files deleted  : 0  Links deleted  : 0
id=0 server: File Content transferred: 4647780 bytes
id=0 server: [Wed Jun 13 10:29:55 2001] Deployment COMPLETED
```

## Source Micro Deployment Log

The *source micro deployment log* contains log entries for the movement of files between the source host and one target host. If the deployment is a fan-out deploying to several target hosts, each source/target deployment will log its own micro deployment log.

The source micro log by default resides in the following location:

```
od-home/log/src.deployment.definition.source-server.to.target-server.log
```

where the following variables apply:

- *deployment* — the name of the associated deployment.
- *definition* — the name of the definition in the deployment configuration that contains the source/target pairing.
- *source-server* — the name of the source host sending the deployment.
- *target-server* — the logical name of the target host server receiving a deployment as it appears in the nodes configuration file of the sending host.

If your deployment was named *monthly*, the definition named *corporate*, your sending host named *mars*, and the target host named *venus*, then the source micro deployment log file name would be:

```
src.monthly.corporate.mars.to.venus.log
```

If your fan-out deployment had following targets:

- *venus*
- *jupiter*

then the sending host would have the two corresponding source micro deployment log files:

```
src.monthly.corporate.mars.to.venus.log
```

```
src.monthly.corporate.mars.to.jupiter.log
```

To access the source micro deployment log from the user interface, follow these steps:

1. Select **Deployment > View Deployment** to display the Source Deployment window.
2. Select the host containing the deployment whose source macro deployment log you want to view from the **Selected Host** list.
3. Select **Sending** from the **View** list. All the deployments that the host sends are displayed in a table.
4. Click the link of the deployment whose source micro deployment log you want to view. The **Details** table appears at the bottom of the window, displaying a separate row for each target host of the deployment.
5. Click **View Log** for the appropriate target host. A separate browser window appears displaying the OpenDeploy Log Viewer window containing the source micro deployment log (Figure 35).

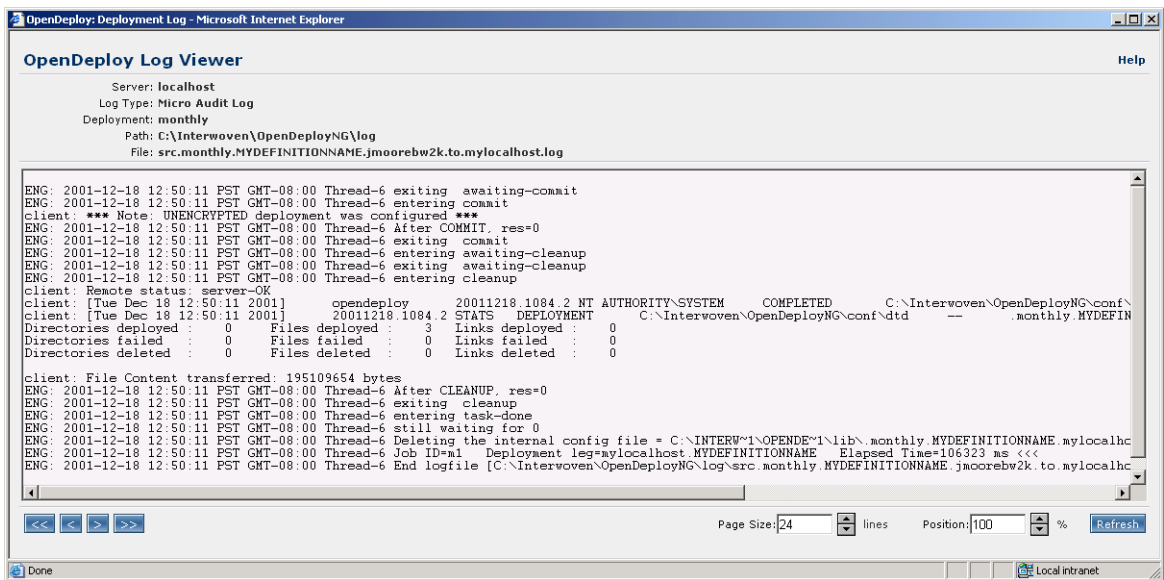


Figure 35: Source Micro Deployment Log

## Receiver Micro Deployment Log

The *receiver micro deployment log* provides a similar service for OpenDeploy hosts receiving deployments as the source micro deployment log does for sending hosts. The receiver micro deployment log contains entries regarding the movement of files between the source and target hosts during the deployment.

The receiver micro log by default resides in the following location:

```
od-home/log/rcv.deployment.definition.source-server.to.target-server.log
```

where the following variables apply:

- *deployment* — the name of the associated deployment.
- *definition* — the name of the definition in the deployment configuration that contains the source/target pairing.
- *source-server* — the name of the source host sending the deployment.
- *target-server* — the logical name of the target host server receiving a deployment as it appears in the nodes configuration file of the sending host.

If your deployment was named *monthly*, the definition named *corporate*, your sending host named *mars*, and the target host named *venus*, then the receiver micro deployment log file name would be:

```
rcv.monthly.corporate.mars.to.venus.log
```

You must select **Receiving** from the **View** list in the Source Deployments window to access micro deployment logs. See “Source Micro Deployment Log” on page 167 for more information.

## Logging Levels

OpenDeploy provides the following logging level options:

- **Verbose** — logs high level of detail on deployment events as they occur. This logging level is best suited for troubleshooting deployment problems or evaluating deployment performance. Verbose logging can create very large log files. This is the default logging level.
- **Normal** — logs standard status and error messages. In most cases, this level of logging provides a sufficient amount of detail to meet your needs.

You can configure logging settings both on an OpenDeploy server basis, and on a deployment configuration basis. See “Logging Configuration Settings” on page 171 for more information.

Settings for deployment logging in the base server or receiver configuration can be overridden in the user interface, or by the deployment configuration. See “Logging Rules Hierarchy” on page 173 for more information.

### Defining Logging Levels in the User Interface

Any time you manually start a deployment from the OpenDeploy user interface (Figure 36), you can specify the level of logging for that deployment. A level specified here automatically overrides any logging levels specified in the base server or deployment configurations.



Figure 36: Log Levels in the User Interface

### Defining Logging Levels from the Command Line

You can specify the logging level for a deployment you start using the `iwodstart` command-line tool by including the `-V` option and the desired logging level. For example:

```
iwodstart deployment -V verbose or
```

```
iwodstart deployment -V normal
```

See “Running Deployments” on page 141 for more information.

## Logging Configuration Settings

You can configure logging in both base server and receiver host configuration files (by default `odbase.xml` and `odrcvr.xml`) and in individual deployment configurations (for example, `test.xml`). Defining the logging settings in the configurations automates the logging rules that apply when a deployment runs. Logging settings are defined in the `logRules` element and its associated attributes.

### Base Server and Receiver Configurations

In the base server and receiver configuration file, the `logRules` element appears as:

```
<logRules maxBytes="x" level="y" directory="z" />
```

where *x*, *y*, and *z* are the values for the following attributes:

- `maxBytes` — specifies the maximum size in bytes a log file is allowed to grow before the file is closed and OpenDeploy begins writing to a new file. This value is known as the *rollover threshold*. The default `maxBytes` value is 32 megabytes. You can specify different byte measurements in the value, including megabytes (mb), kilobytes (kb), and bytes (b). For example:

```
maxBytes="10mb" or
```

```
maxBytes="10000kb" or
```

```
maxBytes="10000000b"
```

indicates that the log file size can grow to 10 megabytes before OpenDeploy will close that log file and start a new one.

Ensure that you include the proper measurement indicator when setting the threshold size. If no recognizable size measurement is indicated, OpenDeploy uses its default value instead. For example, if the following value was specified:

```
maxBytes="10"
```

OpenDeploy would ignore that stated value and use the default value (32mb) instead.

If the unit of measure is present but unrecognized by OpenDeploy, the default value is used. For example, if the following value was specified:

```
maxBytes="1000x"
```

OpenDeploy would ignore this value and use the default value (32mb).

OpenDeploy will not honor a `maxBytes` value of less than 100 kilobytes (100kb). For example, if the following value was specified:

```
maxBytes="50kb"
```

OpenDeploy would ignore this value and use the default value (32mb) instead.

See “Log File Size Management” on page 174 for more information on rollover threshold.

- `level` — indicates the level and type of logging OpenDeploy will perform.
  - `verbose` — logs high level of detail on deployment events as they occur. This logging level is best suited for troubleshooting deployment problems or evaluating deployment performance. Verbose logging can create very large log files. This is the default logging level.
  - `normal` — logs standard status and error messages. In most cases, this level of logging provides a sufficient amount of detail to meet your needs.
- `directory` (base server and receiver configuration only) — specifies the absolute path directory location for log files. The default location is:

```
od-home/log
```

## Deployment Configurations

The `logRules` element functions the same in a deployment configuration as it does in a base server or receiver configuration file, except that the `directory` attribute is not present. For example:

```
<logRules maxBytes="10mb" level="normal" />
```

You can only specify an alternative log file home in the base server or receiver configuration file. Logging settings for macro and micro deployment logs in a deployment configuration will override logging settings in the base server or receiver configuration.

## Logging Rules Hierarchy

The following logging rules hierarchy apply to logging rules:

### Base Server and Receiver Host Logging

The logging levels for the base server and receiver logs are specified in their associated configurations. The level of logging is defined as the value of the `level` attribute in the `logRules` element. Logging levels for this type of logging cannot be overridden. If the `logRules` element or any of its attributes are absent from the base server or receiver configuration, OpenDeploy will use the following default settings:

- `level` — `verbose`
- `maxBytes` — `32 mb`
- `directory` — `od-home/logs`

### Macro and Micro Deployment Logging

The following hierarchy applies to the logging verbosity and maximum file size for deployment macro and micro logs:

- Logging levels specified in the OpenDeploy user interface or the `iwodstart` command-line tool take precedence.
- If the previous parameters are not specified, logging settings in the deployment configuration take precedence.
- If neither of the previous parameters are specified, logging settings in the base server and receiver files take precedence.
- If no other parameters are specified, OpenDeploy will use its default logging settings. See “Logging Configuration Settings” on page 171 for more information.
- If there are any syntax errors in the specified maximum bytes value, such as a unit of measurement being absent or unreadable, OpenDeploy will use its default values for these circumstances. See “Logging Configuration Settings” on page 171 for more information.

## Log File Size Management

Log files can grow large quickly, especially with large or numerous deployments. Using verbose logging (the default logging level) can also generate large log files. You should determine how much storage space to devote to log files before setting the logging type. OpenDeploy uses a log file *rollover threshold* feature to determine the maximum size a single log file can grow before that file is closed to new log entries and a new log file is generated.

The deployment macro logs for the source host and the target hosts are linked for rolling over. When the source host's macro log file requires being rolled over because it has met or exceeded its rollover threshold, the corresponding deployment macro log on the receiving host will also be rolled over, even if it has not reached its rollover limit. The source host determines when a rollover is required.

The deployment micro logs are rolled over in a manner similar to that of macro logs. The source host determines when a log file rollover must occur, and both the source and target micro logs are rolled over together. If a deployment is a fan-out type that includes multiple source/target pairs, the logs of each source/target pair are rolled over independent of other target-source pairs.

### Rollover Threshold Size Determination

The threshold size of the log file is specified in the `logRules` element's `maxBytes` attribute in the base server and receiver configurations files, and in the deployment configurations. If that value is not specified, or if the element is not defined in the configuration, OpenDeploy will look to the same element in the base server configuration file for logging information. If that information is still not present, OpenDeploy will use the default size of 32 MB. See "Logging Configuration Settings" on page 171 for more information.

### Rolled Over Log File Naming

OpenDeploy will roll over a log file when it detects the file size exceeds its specified rollover size as indicated by its `maxBytes` attribute value. A serial naming convention is used to indicate the order of the archived log files. OpenDeploy uses a counter file (*counter.cnt*) to manage the generation of log archive files. Do not move or delete the counter file from the log directory.

When the log file is rolled over, that log's file name is appended with a four-digit extension. This extension starts at 0001 and increases by 1 each time the same log is rolled over. Each log has a separate counter to keep track of rollovers. OpenDeploy subsequently creates a new log file with the original log file name and will start writing log entries to it. For example, if the following log file:

```
src.monthly.log
```

reached its rollover threshold level, OpenDeploy would close this file to further entries and subsequently archive it by adding an appropriate four-digit suffix:

```
src.monthly.log1234
```

In the following example, the log file `src.single.mars.to.venus.log` has been archived four times:

```
4669 Jun 6 10:49 src.single.mars.to.venus.log (current log)
5 Jun 6 10:49 src.single.mars.to.venus.log.cnt (counter)
3877 May 15 15:40 src.single.mars.to.venus.log0001 (first archive)
2126 May 15 15:40 src.single.mars.to.venus.log0002 (second archive)
2126 May 15 15:42 src.single.mars.to.venus.log0003 (third archive)
3901 May 23 13:39 src.single.mars.to.venus.log0004 (fourth archive)
```

When the log rollover extension reaches 9999, the next time it rolls the log over, it will reset to 0001, followed by 0002 and so forth. If the log file with suffix 0001 already exists, that file will be overwritten by the new one as the extension resets. If you want to preserve old log files that are at risk of being overwritten, you should move them to another location.

## Log File Recovery

The following sections explain log file recovery in OpenDeploy.

### Base Server and Receiver Log Files

If the OpenDeploy base server or receiver log file is deleted, OpenDeploy will detect that it is missing and create a new log when one of the following event occurs:

- When you start a deployment manually from the OpenDeploy user interface or using the `iwodstart` command line tool, or if a scheduled deployment is run.
- When you refresh the host through the OpenDeploy user interface or the `iwodserverreset` command-line tool. If the OpenDeploy base server or receiver configuration files have not been changed, this is a convenient way to generate new server log files if the existing ones become lost or damaged.
- When any of the following security related events occur on the OpenDeploy host:
  - The list of users in a role is viewed.
  - A user is added or removed from a role.
  - A deployment is added or removed from an user in the `od-user` role.
  - A user is denied access to an OpenDeploy function.
- When the OpenDeploy host is restarted after having been stopped.

### Deployment Log Files

OpenDeploy will automatically generate new deployment macro and micro log files on both the source and receiver hosts any time existing deployment log files are not detected. If a deployment log file is lost or damaged while that deployment is in progress, no recovery is possible. However, because deployments are logged on both the sending and receiving hosts, you can view the remaining logs.

## Scheduling

You can schedule a deployment to take place any time day or night. You can schedule the deployment to run on a one-time only basis, or recurrently on intervals from a few minutes to monthly. Scheduling deployments frees individuals from having to manually start a deployment. You can schedule deployment to take place at low network traffic periods such as evenings and weekends when they will not interfere with other tasks. You can also schedule a deployment to take place in conjunction with other events, such as a product announcement.

Any individual holding an Administrator role on the OpenDeploy server can schedule any deployment on that host. Individuals with User accounts on an OpenDeploy server can schedule those deployments assigned to them. Individuals holding either an Administrator or User role can view all schedules.

You can schedule deployments using the user interface or from the command line using command-line tools.

## Scheduling from the User Interface

You can create, edit, delete, and view deployment schedules in the OpenDeploy user interface. Creating and editing schedules is performed in the New Schedule window (Figure 37). Here you can specify the time and date the deployment will start. If you want it to occur more than once on a regular basis, such as daily or weekly, you can select that as well. Depending on the frequency level you assign to the scheduled deployment, the New Schedule window will prompt you for additional scheduling information. You can also specify an end date when the schedule is no longer in effect.

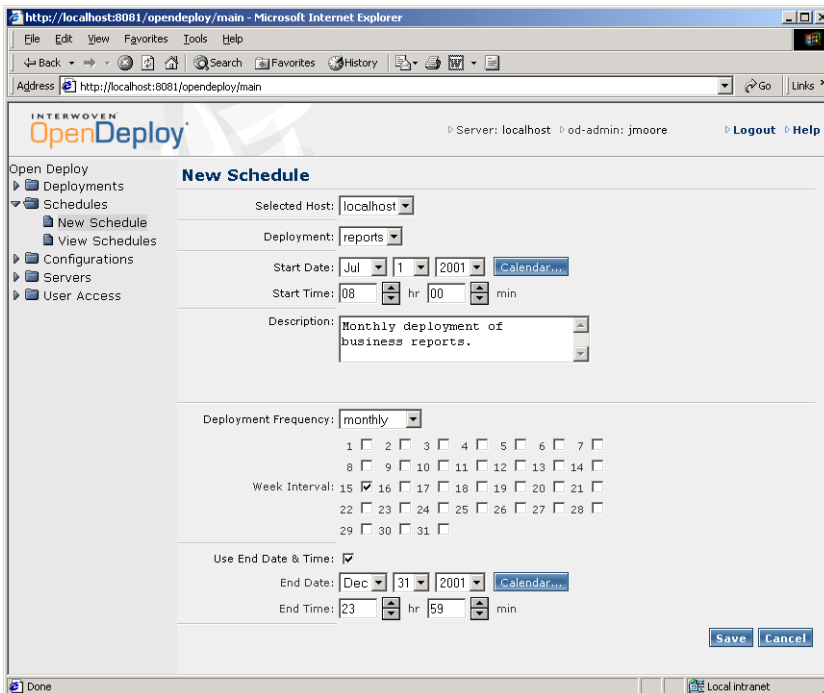


Figure 37: New Schedule Window

## Scheduling Deployments

To schedule a deployment, follow these steps:

1. Select **Schedules > New Schedule** to display the New Schedule window.
2. Select the OpenDeploy server whose deployments you want to schedule from the **Selected Host** list.
3. Select the deployment you want to schedule from the **Deployment** list.
4. Select the month, day, and year on which you want the deployment to start from the **Start Date** lists. You can also click **Calendar** to display a pop-up calendar window. Select the date in this window, and it will automatically be placed in the **Start Date** lists.
5. Select the hour and minute on which you want the deployment to start from the **Start Time** lists. Use the 24-hour clock system, such as 13 to indicate 1 pm.
6. Enter a description of the deployment in the **Description** box. For example:

This is a deployment that updates all our product pages nightly.

7. Select one of the following options from the **Deployment Frequency** list:
  - **Once** — select if the deployment is not recurring.
  - **Sub-hourly** — select to enable deployments recurring in a fixed number of minutes. The **Sub-Hourly** section appears at the bottom on the window (Figure 38). Enter the interval in minutes between deployments in the **Minute Interval** box.
  - **Hourly** — select to enable deployments recurring in a fixed number of hours. The **Hourly** section appears at the bottom on the window (Figure 38). Enter the interval in hours between deployments in the **Hour Interval** box.
  - **Daily** — select to enable deployment recurring in a fixed number of days. The **Daily** section appears at the bottom on the window (Figure 38). Enter the interval in days between deployments in the **Day Interval** box.
  - **Weekly** — select to enable deployment recurring in a fixed number of weeks, and on the same day. The **Weekly** section appears at the bottom of the window (Figure 38). Enter the interval in weeks between deployments in the **Week Interval** box. Select the day of the week the deployment will occur in the **Day of the Week** list.



- **Monthly** — select to enable deployments recurring every month on the same date. The **Monthly** section appears, containing a 31 day calendar (Figure 38). Check each date that the monthly deployment will occur. If you select a date that does not occur every month, for example “31,” then that deployment will not occur until the next month that includes that date. A date of “31” would skip June, but take place in July.

sub-hourly

hourly

daily

weekly

monthly

Figure 38: New Schedules Frequency Features

If you selected any deployment frequency option other than **Once**, continue to the next step. Otherwise, click **Save** to complete the schedule.

8. Check the **Use End Date & Time** box if you want to designate an end date for the recurring deployments. If you do not check this box, the recurring deployments will take place indefinitely.
9. Select the month, day, and year on which you want the deployment to end from the **End Date** lists. You can also click **Calendar** to display a pop-up calendar window. Select the date in this window, and it will automatically be placed in the **End Date** lists.
10. Select the hour and minute on which you want the recurring deployment to end from the **End Time** lists.
11. Click **Save** to complete the new schedule. The View Schedules window appears, displaying the new schedule you just created along with the other scheduled deployments.

## Viewing Deployment Configuration Schedules

Each time you add a schedule, that schedule is displayed in the Deployment Schedules window (Figure 39).

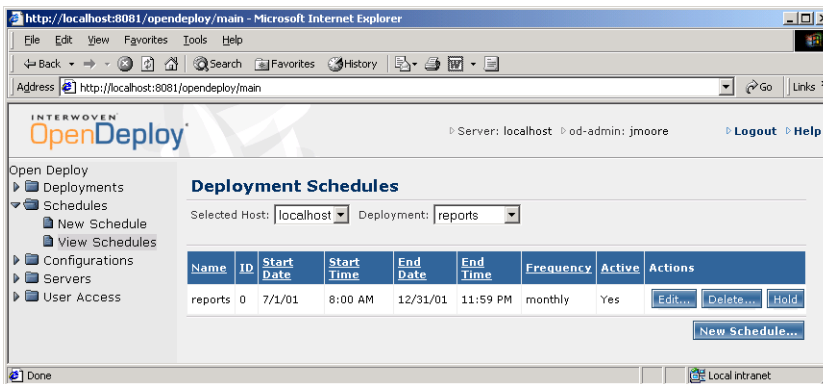


Figure 39: Deployment Schedules Window

This window displays the following information regarding your scheduled deployment:

- Start time and date
- End time and date (if necessary)
- Frequency (if necessary)
- Whether or not it is active

You can also display all the scheduled deployments for an OpenDeploy host by selecting **view all** from the **Deployment** list (Figure 40).

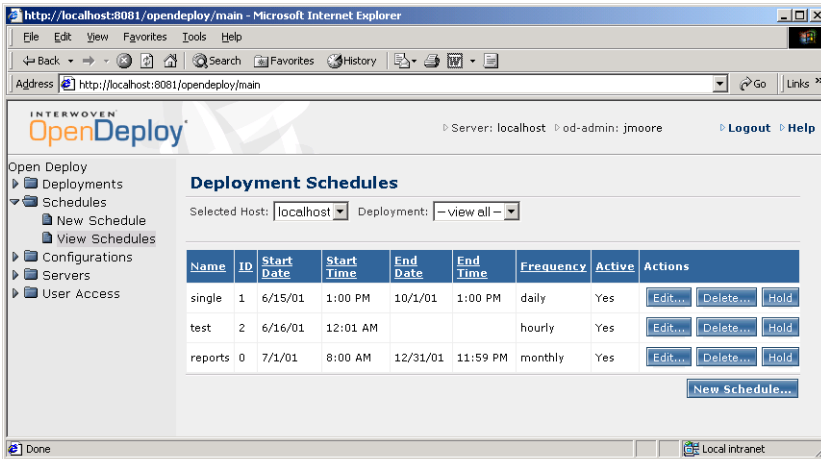


Figure 40: Deployment Schedules Window Displaying All Scheduled Deployments

## Viewing Scheduled Deployment Information

To view a deployment schedule, follow these steps:

1. Select **Schedules > View Schedule** to display the Deployment Schedules window.
2. Select the name of the OpenDeploy server whose deployment scheduling information you want to view from the **Selected Host** list.
3. Select the deployment whose scheduling information you want to view from the **Deployment** list, or select **view all** to display all of them.

The following information is displayed regarding each deployment listed:

- **Name** — displays the name of the deployment.
- **ID** — displays the identification number of the scheduled deployment.
- **Start Date** — displays the day, month, and year specified as the start date when the schedule was added. This may not be the same as the date when the first scheduled deployment will occur.

- **Start Time** — displays the time on the start date specified as the start time when the schedule was added. This may not be the same as the time when the first scheduled deployment will occur.
- **End Date** — displays the day, month, and year specified as the end date when the schedule was added. This may not be the same as the date when the last scheduled deployment will occur.
- **End Time** — displays the time on the end date specified as the end time when the schedule was added. This may not be the same as the time when the last schedule deployment will occur.
- **Frequency** — displays how often the recurring scheduled deployment runs: sub-hourly, hourly, daily, weekly, or monthly.
- **Active** — displays whether or not the scheduled deployment is active.

### Editing Scheduled Deployments

To edit a scheduled deployment, follow these steps:

1. Select **Schedules > View Schedule** to display the Deployment Schedules window.
2. Select the name of the OpenDeploy server whose deployment scheduling information you want to view from the **Selected Host** list.
3. Select the deployment whose scheduling information you want to edit from the **Deployment** list. That scheduled deployment is displayed.

You can also select **view all** to display all the scheduled deployment for the OpenDeploy host.

4. Click **Edit** to display the Edit Schedule window if you want to change any aspect of the existing schedule. The Edit Schedule window looks and functions similarly to the New Schedules window. Here you can change any item of the scheduled deployment.
5. Make you changes and click **Save**.



## Deleting Scheduled Deployments

To delete a scheduled deployment, follow these steps:

1. Select **Schedules > View Schedule** to display the Deployment Schedules window.
2. Select the name of the OpenDeploy server whose deployment scheduling information you want to view from the **Selected Host** list.
3. Select the deployment whose scheduling information you want to edit from the **Deployment** list. That scheduled deployment is displayed.

You can also select **view all** to display all the scheduled deployment for the OpenDeploy host.

4. Click **Delete** to remove the schedule from the scheduler database. You will be prompted to confirm that you want to delete the schedule. If you confirm the deletion, that schedule will be removed from the Deployment Schedules window.

## Activating and Deactivating Scheduled Deployments

When you creating a new schedule, it is automatically activated and will run at it scheduled start date. You can stop the scheduled deployment from occurring without deleting it by deactivating it.

To deactivate a scheduled deployment, follow these steps:

1. Select **Schedules > View Schedule** to display the Deployment Schedules window.
2. Select the name of the OpenDeploy server whose deployment scheduling information you want to view from the **Selected Host** list.
3. Select the deployment whose scheduling information you want to edit from the **Deployment** list. That scheduled deployment is displayed.

You can also select **view all** to display all the scheduled deployment for the OpenDeploy host.

4. Click **Hold** to deactivate that deployment. The **Active** column will display **no** for that scheduled deployment, and the **Hold** button will change to **Activate**.

To reactivate a deactivated scheduled deployment, repeat the same steps you did to deactivate the deployment, and click **Activate**. The **Active** column will display **yes** for that scheduled deployment, and the **Activate** button will change to **Hold**.

## Scheduling from the Command Line

You can use OpenDeploy command-line tools to perform the following scheduling-related tasks:

- Add schedules to deployment configurations.
- Delete existing schedules from deployment configurations.
- Display scheduling information on a selected deployment.
- Activate or deactivate a scheduled deployment.

Scheduling command-line tools only can be issued on the host where the OpenDeploy base server software is installed. These commands can be issued by anyone regardless of whether than hold an Administrator or User role. There are no authentication or authorization checks on individuals issuing these commands.

### Adding a Schedule

To add a schedule to a deployment using the command line, follow these steps:

1. Navigate to the following directory:

```
od-home/bin
```

2. Add a schedule for a deployment by entering the following command at the prompt:

```
iwodschedadd deployment options
```

where *deployment* is the name of the deployment you are scheduling.

There are various options associated with the `iwodschedadd` command-line tool. Here is a listing of these options, along with the usage syntax:

```
iwodschedadd -h | -v
```

```
iwodschedadd deployment [-r [n][m|h|d|w]] [-s [n][m|h|d|w]]  
[-e [n][m|h|d|w]]
```

-h	Displays help information.
-v	Displays version information.
<i>deployment</i>	Name of the deployment being scheduled.
-r	Repeat every <i>N</i> minutes, hours, days, or weeks.
-s [ <i>N</i> ][ <i>m</i>   <i>h</i>   <i>d</i>   <i>w</i> ]	Time from current time to use as start date. The default is 1 minute from current time when the command is entered.
-e [ <i>N</i> ][ <i>m</i>   <i>h</i>   <i>d</i>   <i>w</i> ]	Amount of time from current time to use as end date. The default end time is none. The scheduled deployment will continue indefinitely.
<i>n</i>	A numerical value.
<i>m</i>	Minutes.
<i>h</i>	Hours.
<i>d</i>	Days.
<i>w</i>	Weeks.

### One-Time Only Deployments

If you only want to run the scheduled deployment once, then add the `-n` option to indicate the schedule in non-recurring. In the following example, if you want to schedule the deployment *reports* to deploy a single time a week from now at the same time of day it is currently, you would enter the following command at the prompt:

```
iwodschedadd reports -n -s 1w
```

## Recurring Deployments

If you want your scheduled deployment to run indefinitely at the interval and time you specified, add the `-r` option and the time interval. You can also use the `-s` option and a time period to designate the time of day the deployment will start. Otherwise, the deployment will start at one minute past the time you enter the command. In the following example, if you wanted to schedule the deployment *reports* to run once a day starting at a time one hour from the time you are adding the schedule, you would enter the following command at the prompt:

```
iwodschedadd reports -r 1d -s 1h
```

## Recurring Deployments with End Dates

You can specify an end date on which a recurring deployment will cease by including the `-e` option and the amount of time from now that the recurring deployment will cease. If you do not include an end date, the scheduled deployment will occur indefinitely. In the following example, if you wanted the recurring scheduled deployment from the previous example to cease in two weeks, you would enter the following command at the prompt:

```
iwodschedadd reports -r 1d -s 1h -e 2w
```

## Viewing Scheduled Deployment Information

You can access information on any schedule assigned to your deployment, or all the schedules together, using the `iwodschedget` command-line tool. Several other scheduling-related command-line tools require the schedule ID and other scheduling information that you can get using this tool.

To view information on your deployments schedules, follow these steps:

1. Navigate to the following directory:

```
od-home/bin
```

2. Display the schedule information of a deployment by entering the following command at the prompt:

```
iwodschedget deployment options
```

where *deployment* is the name of the deployment.



There are various options associated with the `iwodshedget` command-line tool. Here is a listing of these options, along with the usage syntax:

```
iwodshedget -h | -v
```

```
iwodshedget -a
```

```
iwodshedget -d deployment
```

```
iwodshedget -o deployment -j ID
```

<code>-h</code>	Displays usage information.
<code>-v</code>	Displays usage information.
<code>-a</code>	Gets all schedules. This is the default option.
<code>-d <i>deployment</i></code>	Gets all schedules for a particular deployment.
<code>-o <i>deployment</i></code>	Gets one schedule. Requires the deployment name and the deployment ID number.
<code><i>deployment</i></code>	The name of the deployment configuration.
<code>-j <i>ID</i></code>	Specifies a job. The ID number of the deployment. Each time a deployment runs, that deployment is given a unique ID number. Similarly, when you schedule a deployment, that scheduled deployment is also given a unique ID number. Use the <code>-a</code> option to see all the ID number for your deployment.

If you wanted to view the schedule information for all of the scheduled deployments residing on your OpenDeploy host, you would enter the following command at the prompt:

```
iwodshedget -a
```

If you wanted to view all schedules for the deployment *reports*, you would enter the following command at the prompt:

```
iwodshedget -d reports
```

If you wanted to view schedule information for the deployment *reports* with an ID number of “2,” you would enter the following command at the prompt:

```
iwodschedget -o reports 2
```

### Deleting a Schedule

To delete a schedule using the command line, follow these steps:

1. Navigate to the following directory:

```
od-home/bin
```

2. Delete a schedule from a deployment by entering the following command at the prompt:

```
iwodscheddelete deployment options
```

where *deployment* is the name of the deployment.

There are various options associated with the `iwodscheddelete` command-line tool. Here is a listing of these options, along with the usage syntax:

```
iwodscheddelete -h | -v |
```

```
iwodscheddelete deployment -j ID
```

```
iwodscheddelete "dep_name_pattern*" [-j ID]
```

`-h` Displays usage information.

`-v` Displays version information.

*deployment* The name of the deployment configuration.

`-j ID` Specifies a job. The ID number of the deployment. Each time a deployment runs, that deployment is given a unique ID number. Similarly, when you schedule a deployment, that scheduled deployment is also given a unique ID number. Use the `iwodschedget -a` command to see all the ID number for your deployment.



<code>"dep_name_pattern"</code>	Deletes schedules based on a wild card name selection, with an optional job identifying number (-j option). The wild card pattern must be quoted ( <code>"sample"</code> ). If the optional job identifying number (-j option) is not present, all scheduled deployments beginning with <code>"dep_name_pattern"</code> will be deleted. If the job identifying number is present, only a scheduled deployment beginning with <code>dep_name_pattern</code> and having a job identifying number equal to the specified value will be deleted.
---------------------------------	---

Because a deployment can have multiple schedules assigned to it, each individual schedule is issued its own unique ID number by OpenDeploy at the time of its creation. You must specify this ID number when you use the `iwodscheddelete` command to ensure that only the schedule you want is being deleted. You can determine this ID value by using the `iwodschedget` command-line tool. See “Viewing Scheduled Deployment Information” on page 187 for more information.

For example, if you wanted to delete a schedule for the deployment `reports` with the ID of “5,” you would enter the following command at the prompt:

```
iwodscheddelete reports 5
```

### Activating and Deactivating a Schedule

When you creating a new schedule, it is automatically activated and will run at its scheduled start date. You can stop the scheduled deployment from occurring without deleting it by deactivating it.

To activate or deactivate a schedule using the command line, follow these steps:

1. Navigate to the following directory:

```
od-home/bin
```

2. Activate or deactivate a scheduled deployment by entering the following command at the prompt:

```
iwodschedactivate deployment options
```

where *deployment* is the name of the deployment.

There are various options associated with the `iwodschedactivate` command-line tool. Here is a listing of these options, along with the usage syntax:

```
iwodschedactivate -h | -v
```

```
iwodschedactivate -a deployment -j ID
```

```
iwodschedactivate -a "dep_name_pattern" [-j ID]
```

```
iwodschedactivate -d deployment -j ID
```

```
iwodschedactivate -d "dep_name_pattern" [-j ID]
```

<code>-h</code>	Displays usage information.
<code>-v</code>	Displays version information.
<code>-a <i>deployment</i></code>	Activates a specific scheduled deployment.
<code>-a "<i>dep_name_pattern</i>"</code>	Activates a scheduled deployment with an optional <i>jobID</i> (-j option) using a wild card pattern format. The wild card pattern must be quoted (" <i>sample</i> *"). If no -j option is present, all scheduled deployments beginning with <i>dep_name_pattern</i> will be changed. If a -j option is present, only a scheduled deployment beginning with <i>dep_name_pattern</i> and having a <i>jobID</i> equal to the job identifying number will be changed.
<code>-d <i>deployment</i></code>	Deactivates a specific scheduled deployment, using the <i>deployment</i> and -j <i>ID</i> options.
<code>-d "<i>dep_name_pattern</i>"</code>	Deactivates a scheduled deployment with an optional job identifying number (-j option), using a wild card format. The selection rules are the same as those stated in the schedule activation description above.
<i>deployment</i>	The name of the deployment configuration.



`-j ID`

Specifies a job. The ID number of the deployment. Each time a deployment runs, that deployment is given a unique ID number. Similarly, when you schedule a deployment, that scheduled deployment is also given a unique ID number. Use the `iwodschedget -a` command to see all the ID number for your deployment.

If you wanted to deactivate the scheduled deployment reports with an ID of “5”, you would enter the following command at the prompt:

```
iwodschedactivate -d reports 5
```

Conversely, to reactivate the deployment, you would enter the following command at the prompt:

```
iwodschedactivate -a reports 5
```

## Uploading Deployment Configurations

The required location for deployment configuration files is:

```
od-home/conf
```

Deployment configuration files you place in this location will appear in the Start Deployment window’s **Deployment** list as a selectable deployment. You can run this deployment from the user interface or the command line, assuming that the deployment conforms to the XML syntax and deployment configuration DTD rules, and that individuals in the User role have the proper access.

You can upload deployment configurations through the Upload Configuration window (Figure 41).

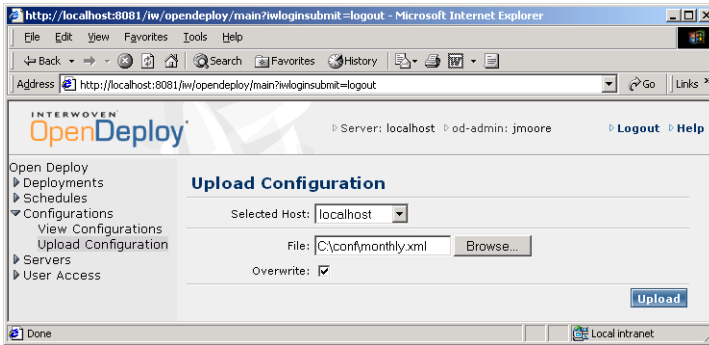


Figure 41: Upload Configuration Window

This feature will copy a configuration from any file system location into the *od-home/conf* directory. You must have Administrator role access to import deployment configurations in this manner. Individuals with User role access will be denied this feature. See “Roles and Authorization” on page 132 for more information.

To upload a deployment configuration, follow these steps:

1. Select **Configurations > Upload Configuration** to display the Upload Configuration window.
2. Select the name of the OpenDeploy server receiving the imported deployment configuration from the **Selected Host** list.
3. Enter the path to the file you want to import. You can also click **Browse** and navigate to the location of the file.
4. Check the **Overwrite** box if you are overwriting an existing deployment configuration file.
5. Click **Upload**. The file you uploaded is now available for use.



## Chapter 4

# Deployment Configurations

---

This chapter describes how to configure the deployment configuration file to perform a variety of sophisticated and complex deployments utilizing a range of OpenDeploy features.

## Deployment Configuration Files

A deployment configuration file is an XML-based file containing elements and attributes that define how the deployment will work. Some elements and attribute values are required, while others are optional. In most cases, if an optional value is not specified in the deployment configuration file, OpenDeploy will use a built-in default value. In some cases, OpenDeploy will look to the host's base server or receiver configuration files for setting information if none exists in the deployment configuration file. The rest of this chapter discusses features available in OpenDeploy by modifying the deployment configuration file. You should have some knowledge of XML before modifying these files.

## Understanding the Configuration DTDs

OpenDeploy configuration files are XML files based on the rules defined in the corresponding configuration DTD. Your XML-based configuration files must conform to the rules set forth in these DTDs.

### Elements

Each DTD file consists of various *elements* that provide some function or task. These elements are contained within angled brackets (“< >”) and form the basis of the DTD. In the source code, each element is declared in the following way:

```
<!ELEMENT element_name>
```

In many cases, a given element will have subordinate elements, known as *child elements*. The child elements associated with an element are contained within parentheses following the parent element's declaration. For example:

```
<!ELEMENT element_name (child_element_name)>
```

If an element in the source code has a child element specified, the information for that child element can be found later in the code. An element can have more than one child element. Child elements that are only separated by a space can occur in any order within the parent element. For example:

```
<!ELEMENT element_name (child_element_name1 child_element_name2)>
```

If child elements are separated by a comma (","), then those child elements must appear in that order within the parent element in the code. For example:

```
<!ELEMENT element_name (child_element_name1, child_element_name2)>
```

If child elements are separated by a pipe ("|"), then only one of the child elements listed can be used. For example:

```
<!ELEMENT element_name (child_element_name1 | child_element_name2)>
```

In some cases, there can be restrictions or requirements placed on the usage of elements. Certain symbols placed immediately after an element name indicate these restrictions and requirements. For example:

- `<element_name>` (no symbol) — the element must appear just once.
- `<element_name?>` — the element can be omitted or can occur just once.
- `<element_name*>` — the element can be omitted or can appear one or more times.
- `<element_name+>` — the element must appear at least once, but can occur more than once as well.

## Attributes

Element tags can include one or more optional or mandatory *attributes* that give further information about the elements they support. Attributes only can be specified within the element tag. The presence of an attribute within an element tag requires a corresponding value enclosed in quotes. For example:

```
<element_name attribute="value">
```

The type of value (a number, yes|no, a path) can vary depending on the type and nature of the attribute and element. The documentation for a given attribute will specify the types of values permitted.

An element can have any number of attributes. Multiple attributes in an element tag follow one after another, with no separators. For example:

```
<element_name attribute_name1="value" attribute_name2="value">
```

The attribute declaration always immediately follows its corresponding element declaration. In the source code of the OpenDeploy configuration DTD files, the attributes of a given element are declared in the following way:

```
<!ELEMENT Element_name>
<!ATTLIST Element_name
    attribute_name1      attribute1_type      attribute1_keyword
    attribute_name2      attribute2_type      attribute2_keyword
>
```

The following sections describe attribute types and attribute keywords.

### Attribute Type

The attribute type specifies the type of value that can be associated with the attribute. There are a variety of different attribute types possible. Some are immediately intuitive, such as (yes|no), where the choice is one of the values specified. Others require a more substantial explanation. Here is a list of commonly found attribute types within configuration DTDs:

- **CDATA** — character data such as a text string. CDATA is not recognized as markup language code.
- **ID** — an identifier for the element. No two elements can have the same ID attribute value in the same DTD. These types are usually unique identification numbers or strings.



- IDREF — a pointer to an ID reference. The value must match the value of an ID type attribute that is declared somewhere else in the same DTD.

There are other attribute types possible as well. Consult a book on XML for more information.

### Attribute Keyword

The attribute keyword specifies action the server should take if an associated attribute is left out. One of the following values is used for the attribute keyword:

- #REQUIRED — the attribute is required and should be present. If it is missing, the configuration file is invalid and the application will not work.
- #IMPLIED — the attribute is not required. If the attribute has no value specified, the application will make a suitable substitution.
- #FIXED — the attribute value is fixed at a preset value. No modification to the attribute value is allowed.

## Encoding

The encoding for the nodes configuration file can be encoding other than UTF-8. For example, if a value in the file contains Japanese characters, the encoding will need to be:

```
<? xml version="1.0" encoding="SHIFT_JIS" ?>
```

For French and German, the encoding value would be:

```
<? xml version="1.0" encoding="ISO-8859-1" ?>
```

Check what the appropriate value is for any non-ASCII characters and modify the nodes configuration file encoding as needed. If no encoding is specified, UTF-8 will be used by default.

## Naming Deployment Configurations

Deployment configuration file names can be any length up to the limit supported by the host operating system.

Deployment configurations residing on UNIX hosts cannot have spaces in the file names. Those running on Windows hosts may contain spaces.

All deployment configuration file names must have the `.xml` extension. The file cannot be read by OpenDeploy without this extension.

## Deployment Configuration Structure

All OpenDeploy deployment configurations follow the same structure. All deployment configurations begin at the root element `deploymentConfiguration`. Within the `deploymentConfiguration` element are the following child elements:

- `localhost` (required) — specifies the deployment host name.
- `replicationFarmSet` (required) — the container element for target hosts and host groupings
- `definition` (required) — contains the source/target pairings, including the source and target file locations, and the type of deployment being performed
- `deployment` (required) — contains deployment features, including transactional and Deploy and Run
- `logRules` (optional) — specifies deployment logging settings. If no log rules are specified in the deployment configuration, OpenDeploy will reference the base server and receiver configuration files for logging instructions. If no logging information is present in those files either, OpenDeploy will use its default log settings. See “Logging” on page 155 for more information.



The following example shows a simple deployment configuration using the minimum amount of information:

```
<?xml version="1.0" encoding="UTF-8"?>

<deploymentConfiguration>

    <localNode host="mars.mycompany.com"/>

    <replicationFarmSet>
        <replicationFarm name="MYFARMNAME">
            <nodeRef useNode="MyLocalHost"/>
        </replicationFarm>
    </replicationFarmSet>

    <definition name="MYDEFINITIONNAME">
        <source>
            <sourceFilesystem area="C:\Interwoven\OpenDeployNG\conf\dtd">
                <pathSpecification>
                    <path name="."/>
                </pathSpecification>
            </sourceFilesystem>
        </source>

        <target useReplicationFarm="MYFARMNAME">
            <comparisonRules dateDifferent="yes"/>
            <permissionRules file="0644" directory="0755"/>
            <targetFilesystem area="C:\Interwoven\OpenDeployNG\tmp"/>
        </target>
    </definition>

    <deployment transactional="no">
        <execDeploymentTask useDefinition="MYDEFINITIONNAME"/>
    </deployment>

    <logRules maxBytes="32Mb" level="verbose"/>

</deploymentConfiguration>
```

OpenDeploy will look to the base server (by default `odbase.xml`) or receiver (by default `odrcvr.xml`) configuration files, for direction if no configuration information for a particular feature or function is present in the deployment configuration. If there is no configuration information in these files either, OpenDeploy will either use its built-in settings, or ignore the feature.

Depending on the nature and complexity of the deployment, optional elements and attributes can be added as necessary. In the following example, a variety of additional elements and attributes have been added to the previous example. However, the basic structure remains the same:

```
<?xml version="1.0" encoding="UTF-8"?>

<deploymentConfiguration>

    <localNode host="mars.mycompany.com"/>

    <replicationFarmSet>
        <replicationFarm name="fan-out">
            <nodeRef useNode="a"/>
            <nodeRef useNode="b">
                <nextDeployment deployment="tier2" invokeOnSuccess="yes"/>
            </nodeRef>
            <nodeRef useNode="c"/>
        </replicationFarm>
    </replicationFarmSet>

    <definition name="enable_production">
        <source>
            <sourceFilesystem area="C:\Interwoven\OpenDeployNG\conf\dtd">
                <pathSpecification>
                    <path name="."/>
                    <filters>
                        <excludePath subPath="out/exes"/>
                        <excludePattern regex=".obj$"/>
                    </filters>
                </pathSpecification>
            </sourceFilesystem>
        </source>

        <target useReplicationFarm="fan-out">
            <comparisonRules dateDifferent="yes"/>
            <transferRules doDeletes="yes"/>
            <permissionRules file="0644"
                            directory="0755"
                            user="webmaster"
                            group="webgroup"
                            />
            <targetFilesystem area="C:\Interwoven\OpenDeployNG\tmp"/>
        </target>
    </definition>
```



```
<deployment transactional="no">
  <execDeploymentTask useDefinition="enable_production">
    <deployNRun>
      <dnrFile location="target" when="before" state="success"
        mask=".html$">
        <script cmd="C:\bin\email_admin.bat -user rroe@mycompany.com"
          as="webmaster"
          where="C:\temp"
          async="no"
        />
      </dnrFile>
      <dnrDir location="target" when="after" state="failure"
        mask="images">
        <script cmd="C:\bin\email_admin.bat -user rroe@mycompany.com"
          as="webmaster"
          where="C:\temp"
          async="no"
        />
      </dnrDir>
      <dnrDeployment location="source" when="after" state="success">
        <script cmd="C:\bin\email_admin.bat -user jdoe@mycompany.com"
          as="webmaster"
          where="C:\temp"
          async="yes"
        />
      </dnrDeployment>
    </deployNRun>
  </execDeploymentTask>
</deployment>

<logRules maxBytes="32Mb" level="verbose"/>

</deploymentConfiguration>
```

## Specifying the Deployment Host

The deployment host is the OpenDeploy server that will serve as the source host of the deployment. The deployment configuration must reside on the host that will start the deployment. The deployment configuration cannot exist remotely from the source host. The deployment host is specified as the `localNode` element's `host` attribute value. For example:

```
<deploymentConfiguration>
  <localNode host="mars.mycompany.com" />
  ...
</deploymentConfiguration>
```

This value must be the OpenDeploy host's fully qualified DNS host name or IP address. It cannot be the logical name of the host. This host must also be listed in the host's nodes configuration file (by default `odnodes.xml`).

In some cases you might compose or modify a deployment configuration on your local computer, but intend to move it to an OpenDeploy base server host for actual usage. In these cases, the `host` attribute value must be that of the OpenDeploy base server host.

## Target Replication Farms

*Target replication farms* are groupings of OpenDeploy target host nodes under a single named element. All target hosts listed in a replication farm will receive the same set of deployed files, unless any overrides are specified. Replication farms allow you to simplify the deployment process by deploying a single set of files to a large number of targets without having to individually configure each one.

Each deployment configuration must have at least one target replication farm, even if that farm consists of only a single target host node. You can have as many additional replication farms as you want, as long as each one is uniquely named. An individual target host node can belong to more than one replication farm.

Each target replication farm is designated by the presence of a `replicationFarm` element. The `replicationFarm` element's `name` attribute must contain a unique name. Within each `replicationFarm` element are `nodeRef` child elements. You must have a corresponding `nodeRef` element for each target host node to which you want to deploy files. Each `nodeRef` element must also correspond to a node entry in the nodes configuration file (by default `odnodes.xml`). The `nodeRef` element's `useNode` attribute value must be the same as the target host node's logical name (specified as the node element's `name` attribute value in the nodes configuration file).

Each individual `replicationFarm` element is contained within a single `replicationFarmSet` element for the deployment configuration. In the following example, the target replication farms *europe* and *asia* are specified within the deployment configuration:

```
<deploymentConfiguration>
  <localNode host="mars.mycompany.com" />
  <replicationFarmSet>
    <replicationFarm name="europe">
      ...
    </replicationFarm>
    <replicationFarm name="asia">
      ...
    </replicationFarm>
  </replicationFarmSet>
  ...
</deploymentConfiguration>
```

Within the target replication farm *europe*, the individual target host nodes making up the replication farm are listed, for example:

```
<replicationFarm name="europe">
  <nodeRef useNode="england" />
  <nodeRef useNode="france" />
  <nodeRef useNode="spain" />
</replicationFarm>
```

These target host nodes also must be listed in the nodes configuration file for the deployment host, for example:

```
<nodeSet name="od_receiver_nodes">
  <node name="england" host="london.mycompany.com" port="20014" />
  <node name="france" host="paris.mycompany.com" port="20014" />
  <node name="spain" host="madrid.mycompany.com" port="20014" />
</nodeSet>
```

## Definitions

A *definition* is a uniquely-named pairing of the source file location on the sending OpenDeploy host with one or more target file locations on the receiving hosts. A definition is specified in the deployment configuration with the `definition` element. Its unique name is the value of the `name` attribute. Within the definition element are the `source` and `target` child elements. For example:

```
<deploymentConfiguration>
  ...
  <definition name="webfiles">
    <source>
      ...
    </source>
    <target ...>
      ...
    </target>
  </definition>
  ...
</deploymentConfiguration>
```

Each deployment configuration must have at least one definition. You can have as many additional definitions as you want, as long as each one is uniquely named.

## Source File Location

The source file location is where the source files participating in a deployment reside. This location can either be a file system location or a TeamSite area (if TeamSite is also installed on your OpenDeploy source host). If the deployment is comparison-based, the source file location participates in the file comparison, either with a file system location on the target host, or with another TeamSite area on the source host. If the deployment is file list-based, those files that are deployed must reside in the source file location.

The source file location is defined within the definition by the `source` element, and its `sourceFilesystem` (for directory comparison and file list deployments) and `sourceTeamsite` (for TeamSite comparison deployments) child elements. For example:

```
<definition name="webfiles">
  <source>
    <sourceFilesystem area="C:\website\files" >
      ...
    </sourceFilesystem>
  </source>
  ...
</definition>
```

You can have multiple source file locations configured within the definition. For example:

```
<definition name="webfiles">
  <source>
    <sourceFilesystem area="C:\website\files" >
      ...
    </sourceFilesystem>

    <sourceFilesystem area="C:\images" >
      ...
    </sourceFilesystem>
  </source>
  ...
</definition>
```

However, you should take precautions to avoid overwriting the deployed files from one source file location with those of another.

All your source file locations within a definition must be either a filesystem location or TeamSite area. You cannot combine both types within a single definition.

## Target File Location

The target file location is defined within the definition by the `target` element. Any target file locations listed in a definition are assumed by OpenDeploy to apply to all the target hosts listed in the target replication farms, unless overridden. Only a single target file location is allowed in each definition.

The `useReplicationFarm` attribute specifies the target replication farm on whose target host nodes the target file location will receive the deployed files. The target replication farm value must match an entry present in the source host's nodes configuration file (by default `odnodes.xml`). See "Target Replication Farms" on page 203 for more information.

The target file location must be a file system location. The target hosts are not listed in the definition, only the target file location.

```
<definition name="webfiles">
  ...
  <target useReplicationFarm="MYFARMNAME">
    <targetFilesystem area="D:\website\files" />
  </target>
</definition>
```

TeamSite areas are not supported in the target file location. However, if you want to deploy files to a TeamSite workarea, you can use the TeamSite area virtual path as the value for the target file location. For example:

```
<targetFilesystem area="Y:\default\main\dev\WORKAREA\jdoe">
```

## File Filters and Rules

Also included in a definition are all the rules and filters that determine what files can be deployed by the sending host and received by the target hosts. These include the following:

- File path exclusion filters
- File name pattern exclusion filters
- File comparison rules
- File transfer rules
- File permission rules
- Transfer rules for symbolic links
- Target override rules

See Chapter 5, “Advanced Features” for more information on these features, and how to configure them in your deployment.

## Deployment Tasks

The `deployment` element contains attributes and child elements that allow you to configure the following items:

- Transactional functionality
- Definition-specific configurations for down-rev deployments and Deploy and Run scripting

### Transactional

The `deployment` element provides the ability to make the deployment configuration transactional. If a deployment with the transactional feature enabled fails to successfully deploy all the appropriate files to the target hosts, OpenDeploy will roll back the deployment and restore the target host’s file locations to their previous state. See “Transactional Deployments” on page 229 for more information.

You can enable this feature by assigning a value of `yes` to the `transactional` attribute, for example:

```
<deploymentConfiguration>
  ...
  <deployment transactional="yes">
    ...
  </deployment>
</deploymentConfiguration>
```

## Definition-Specific Configurations

Within the `deployment` element you can configure certain functionality on a definition-specific basis. You can specify a particular definition through the `execDeploymentTask` element. Each `deployment` element must have at least one `execDeploymentTask` child element. If there is only one definition in the deployment configuration, the `execDeploymentTask` element's `useDefinition` attribute value must be that definition's name. For example:

```
<deployment transactional="yes">
  <execDeploymentTask useDefinition="europe" />
</deployment>
```

If you have multiple definitions in a deployment configuration, then you can specify one, some, or all of those definitions with a separate instance of the `execDeploymentTask` element. For example:

```
<deployment transactional="yes">
  <execDeploymentTask useDefinition="europe">
    ...
  </execDeploymentTask>
  <execDeploymentTask useDefinition="asia">
    ...
  </execDeploymentTask>
</deployment>
```

Within the `execDeploymentTask` element, you can configure the following features:

- Downward revision deployments
- Deploy and Run scripts



## Downward Revision Deployments

You can enable OpenDeploy to deploy files specified in a particular definition to deploy to downward revision (OpenDeploy release 4.5.2) target hosts.

This feature is enabled by assigning the value 4.5.2 for the downRev attribute, for example:

```
<deploymentConfiguration>
  ...
  <deployment transactional="yes">
    <execDeploymentTask useDefinition="europe" downRev="4.5.2" />
  </deployment>
</deploymentConfiguration>
```

See “Deploying to OpenDeploy Downward Revision Targets” on page 89 for more information.

## Deploy and Run

Deploy and Run allows you to configure OpenDeploy to execute an external script at a specified stage of the deployment. This stage can be the deployment of a particular type or class of file or directory, or even the success of the deployment itself. See “Deploy and Run” on page 267 for more information.

Deploy and Run scripting is assigned on a definition-specific basis within the `execDeploymentTask` element. The `deployNRun` child element is the container for Deploy and Run configuration. For example:

```
<execDeploymentTask useDefinition="europe" downRev="4.5.2">
  <deployNRun>
    ...
  </deployNRun>
</execDeploymentTask>
```

## Directory Comparison Deployments

During a *directory comparison*, OpenDeploy compares the directories and files on the source host with another set of files and directories residing on the target host. The files and directories that meet the deployment criteria as a result of this comparison can then be moved to the target host.

In Figure 42, the source host *mars* has a file system area defined as:

```
C:\dev\website\files
```

The target host *venus* has a file system area defined as:

```
D:\website\files
```

The deployment configuration file specifies the rules for determining which files should be deployed. It also specifies any additional rules or actions that are applied to the deployed files. OpenDeploy uses these rules during the comparison and deployment phases of its operation. See “Deployment Types” on page 33 for more information.

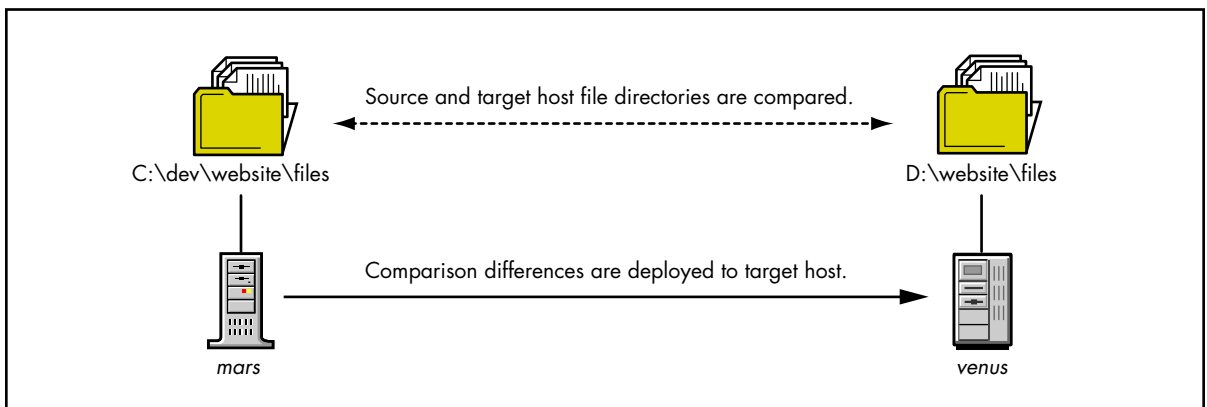


Figure 42: Directory Comparison

## Defining the Source Host File Area

Directory comparison deployments are determined by the `sourceFilesystem` element. This element contains the `area` attribute where the directory containing the source host's files is specified. The `area` attribute value specifies the absolute path for a file system containing the host files. For source hosts, this indicates the path where the files currently reside. For example:

```
area="/website/files" or
```

```
area="C:\website\files"
```

## Specifying a Location Within the Source File System Area

There may be times when you want to identify a particular location within the specified area for a deployment. You can specify locations within the source area by configuring additional elements and attributes within the `source` element.

The `pathSpecification` and `path` elements allow you to specify a particular relative location within the designated source area. In the following example:

```
<source>
  <sourceFilesystem area="C:\website\files">
    <pathSpecification>
      <path name="monthly" />
    </pathSpecification>
  </sourceFilesystem>
</source>
```

the file system area is specified as `C:\website\files`. In order to further specify the subdirectory `monthly` within the area, the `path` element's `name` attribute value was specified as `monthly`.

The `name` attribute is the directory relative to the area specified in the `sourceFilesystem` attribute for your source host. If you want to specify a location multiple levels below the area, you can enter the path to that location relative to the area. For example, if you want to specify the source location `monthly\january` relative to the area, the `path` element would be:

```
<path name="monthly\january" />
```

The `pathSpecification` and `path` elements are required in the deployment configuration file. However, if you do not want to specify a source host location any narrower than the specified area, you can simply list the `path` element's `name` attribute as `"."`, which indicates the area location. This is the default setting for all the OpenDeploy sample configuration files. The following example shows a simple `source` element where no source host location is specified beyond the area:

```
<source>
  <sourceFilesystem area="/website/files">
    <pathSpecification>
      <path name="." />
    </pathSpecification>
  </sourceFilesystem>
</source>
```

## Defining the Target Host Area

The target host area for a directory comparison deployment is specified by the `targetFilesystem` element. This element and its `area` attribute specify the host location where the target files reside after the deployment. Like the `sourceFilesystem` element, the `targetFilesystem` element contains an `area` attribute where you must specify an absolute path to the target host file location. For example:

```
<targetFilesystem area="D:\website\files" />
```

After the `sourceFilesystem` and `targetFilesystem` elements are defined, the directory comparison can take place. The following example below shows the pairing between the source and target host file system-based locations:

```
<source>
  <sourceFilesystem area="/website/files">
    ...
  </sourceFilesystem>
</source>

<target useReplicationFarm="MYFARMNAME">
  <targetFilesystem area="D:\website\files" />
</target>
```

## TeamSite Comparison Deployments

*TeamSite comparison* deployments are specified by the `sourceTeamsite` element and its attributes. The `area` attribute's value is the path of the TeamSite area containing the source host's files. In a TeamSite comparison, the source host files are located in a TeamSite area and are compared with a second TeamSite area in the same backing store on a host with TeamSite installed. The differences between the two TeamSite areas are what is deployed to the target host. You can configure OpenDeploy to compare two TeamSite areas in any single backing store on a multi-backing store TeamSite host.

This type of comparison is most effective if the second TeamSite area contains a mirror image of the files on the eventual target host. This differs from a directory comparison deployment, where files on the source host are compared with a corresponding file location on the target host.

In Figure 43, the source host *mars* is a running TeamSite software in addition to OpenDeploy.

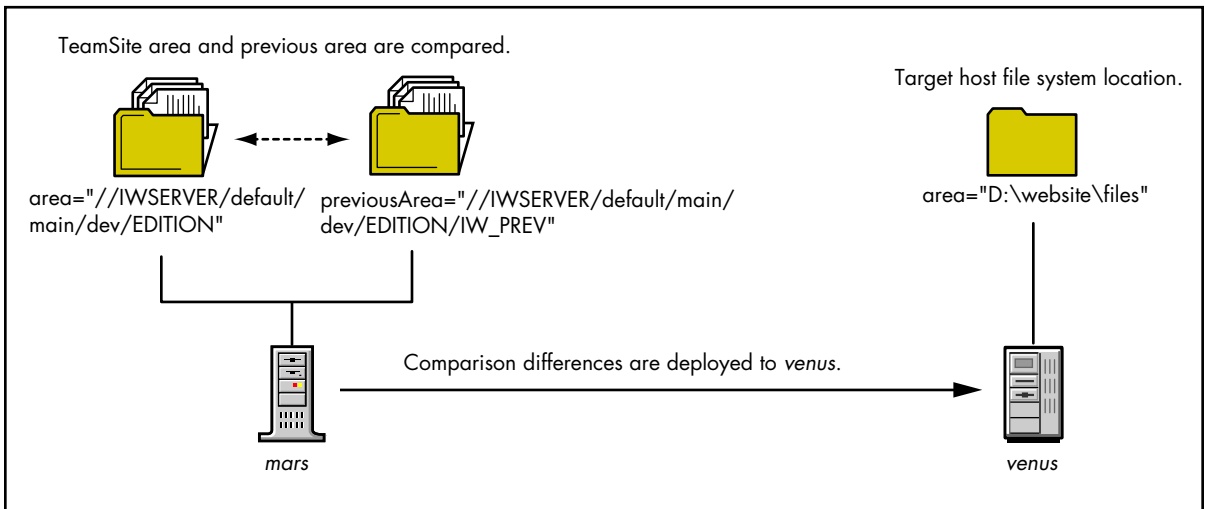


Figure 43: TeamSite Comparison Deployment

It has the most recent TeamSite edition as the value for the `area` attribute:

```
//IWSERVER/default/main/dev/EDITION
```

This area is known as the *primary area* and its TeamSite location is the value for the `area` attribute.

The previous version of this edition represents a mirror image of the files residing on the eventual target host *venus*:

```
//IWSERVER/default/main/dev/EDITION/IW_PREV
```

This previous version is known as the *previous area* and its TeamSite location is the value for the `previousArea` attribute. OpenDeploy compares the two TeamSite areas specified in the `area` and `previousArea` attributes, and determines which files in the more recent edition need to be deployed to the target host. See “TeamSite Comparison” on page 34 for more information on TeamSite comparison-based deployments.

## Defining the Source Host TeamSite Areas

The areas comprising a TeamSite comparison deployment are specified by the `sourceTeamsite` element. The `sourceTeamsite` element contains two attributes:

- `area` — defines the primary TeamSite area
- `previousArea` — defines the previous TeamSite area

### Specifying the Primary TeamSite Area

The `area` attribute defines the TeamSite host area where the new files to be compared are located. This area is typically an edition or a staging area. For example:

```
//IWSERVER/default/main/dev/STAGING
```

Although you can deploy files from a workarea, it is not recommended, because files in a workarea do not have file versioning and other TeamSite content management features applied to them. It is preferred to submit files from a workarea to the staging area, and to perform the deployment from there.



To designate this attribute as the most recently published TeamSite edition, enter the TeamSite path ending simply with “EDITION.” OpenDeploy will automatically access the most recently published edition path. For example:

```
//IWSERVER/default/main/dev/EDITION
```

To specify a another edition for the `area` attribute, enter the complete TeamSite path to that edition. For example:

```
//IWSERVER/default/main/dev/EDITION/ed01022001
```

### Specifying the Previous Area for Comparison

The `previousArea` attribute defines the TeamSite host area against which the content in the `area` attribute is compared. The content of the `previousArea` should be a mirror image of the content contained on the target host receiving the deployed files.

To designate this attribute as the edition that immediately preceded the most recently published one (as specified by the use of `EDITION` in the `area` value), enter the TeamSite path ending simply with `EDITION/IW_PREV`. OpenDeploy will automatically access the next-to-most recently published edition path. For example:

```
//IWSERVER/default/main/dev/EDITION/IW_PREV
```

To specify another edition for the `previousArea` attribute, enter the complete TeamSite path to that edition. For example:

```
//IWSERVER/default/main/dev/EDITION/ed01012001
```

The two designated TeamSite areas within the `sourceTeamsite` element are compared during the TeamSite comparison, and the files that meet the deployment criteria are deployed to the target host.

In the following example:

```
<sourceTeamsite
  area="//IWSERVER/default/main/dev/EDITION"
  previousArea="//IWSERVER/default/main/dev/EDITION/IW_PREV"
  >
  ...
</sourceTeamsite>
```

the last two TeamSite editions are compared with each other, and the differences deployed to the target host.

In some cases, you might want to deploy the entire contents of a TeamSite area. You can perform this task by specifying a TeamSite area that contains no files as the value for the `previousArea` attribute, such as the initial edition that is generated for the TeamSite base branch:

```
<sourceTeamsite
  area="//IWSERVER/default/main/dev/EDITION"
  previousArea="//IWSERVER/default/main/EDITION/INITIAL"
  >
  ...
</sourceTeamsite>
```

### Specifying a Location Within the Source TeamSite Area

You can specify narrower locations within TeamSite areas in the same manner as you can with file system-based areas. The elements and attributes for specifying a location within a TeamSite area-based source host area are essentially the same as for a file system-based source host. The location is relative to the area specified in the `sourceTeamsite` element. However, in a TeamSite comparison deployment, the source host area is specified as a TeamSite area.

You can use the `pathSpecification` and `path` elements and their attributes to specify a location within both TeamSite areas being compared (as indicated by the values of the `area` and `previousArea` attributes), just as you did with file system-based deployment source hosts. The relative directory or path you specify in the `path` element's `name` attribute applies equally to the `area` and `previousArea` locations. You cannot specify a narrowed location on only one of the two areas.

In the following example:

```
<source>
  <sourceTeamsite
    area="//IWSERVER/default/main/dev/EDITION"
    previousArea="//IWSERVER/default/main/dev/EDITION/IW_PREV"
  >
    <pathSpecification>
      <path name="monthly" />
    </pathSpecification>
  </sourceTeamsite>
</source>
```

the two TeamSite areas being compared are the current edition:

```
//IWSERVER/default/main/dev/EDITION
```

and the previous edition:

```
//IWSERVER/default/main/dev/EDITION/IW_PREV
```

By specifying the value “monthly” for the path element’s name attribute, the two TeamSite areas being compared are narrowed to the directory *monthly* located in each TeamSite area.

The two TeamSite areas being compared are now effectively the following:

area — //IWSERVER/default/main/dev/EDITION/monthly *and*

previous area — //IWSERVER/default/main/dev/EDITION/IW\_PREV/monthly

## Defining the Target Host Location

Unlike a directory comparison where the target host is an integral part of the comparison, in a TeamSite comparison the target host simply receives the files the source host determines are appropriate. The target host area for a TeamSite comparison deployment can only be a file system location.

Use the `targetFilesystem` element to designate this type of recipient host location. For example:

```
<targetFilesystem area="C:\website\files"> or  
<targetFilesystem area="/website/files">
```

To deploy files to a TeamSite workarea on your target host, use the file system location rather than the TeamSite path in your configuration. For example:

```
<targetFilesystem area="Y:\default\main\dev\WORKAREA\jdoe">
```

## Use With Deploy and Run Scripts

Evaluation of the `area` and `previousArea` attribute values in TeamSite comparison deployments with respect to the latest and next-to-latest editions, occurs *before* the running of any Deploy and Run scripts. See “Deployment-Based” on page 272 for more information.

## File List Deployments

*File list* deployments do not compare source or target files, either in a file system location or a TeamSite area. Instead, OpenDeploy simply moves the files from the source host to the target host based on the files and directories indicated in the file list itself. Only one list of files per deployment is allowed. All deployment criteria and functionality can be applied to file list deployments, with the exception of the `doDeletes` option.

Unlike directory comparison and TeamSite comparison types of deployments, a file list deployment does not have its own unique element that defines the deployment. Instead, the file list deployment is indicated by the presence of the `filelist` attribute and its value in the `sourceFilesystem` element.

In Figure 44, the source host *mars* is performing a file list deployment to the target host *venus*.

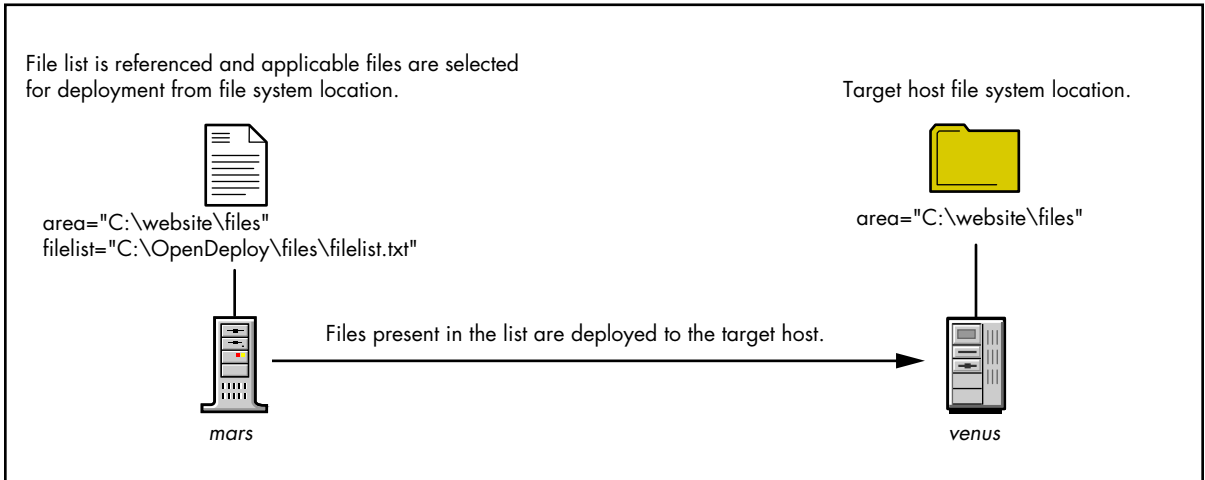


Figure 44: File List Deployment

The source host references the file specified by the `filelist` attribute in the `sourceFilesystem` element:

```
C:\OpenDeploy\files\filelist.txt
```

and proceeds to deploy the files listed there to the target host. The value specified in the `area` attribute:

```
C:\website\files
```

defined as the location of the files to be deployed from the source host, is the same as in a directory comparison deployment.

## Specifying the File List

File list deployments are specified in the `sourceFilesystem` element's `filelist` attribute. The `filelist` attribute specifies the absolute path to the file list file. For example:

```
<sourceFilesystem
  area="C:\website\files"
  filelist="C:\OpenDeploy\files\filelist.txt"
>
...
</sourceFilesystem>
```

The `filelist` attribute and its value within the `sourceFilesystem` element must be present for OpenDeploy to recognize the configuration as a file list deployment. Without it, the deployment is performed as a directory comparison.

## Editing the File List

The file list is a text file that you can create and modify using your favorite text editor. This list contains a series of entries of files and directories that are relative to the `area` attribute value specified in the `sourceFilesystem` element. Here is an example of file list entries:

```
www/index.html
www/andre/index.html
www/products.html
```

You must follow the path syntax of your source host computer type when editing the file list. Forward slashes (“/”) can be used for either Windows or UNIX hosts:

```
www/andre/index.html
```

while backslashes (“\”) are only permitted on Windows hosts:

```
www\andre\index.html
```



Because these files and directories are relative to the specified file system, you do not need to enter the absolute path of each entry in the file list. Instead, just enter the path relative to that `area` attribute you specified in the `sourceFilesystem`. For example, if you had specified the `sourceFilesystem` element's `area` attribute as having the following value:

```
<sourceFilesystem
  area="C:\website\files"
  filelist="C:\OpenDeploy\files\filelist.txt"
>
...
</sourceFilesystem>
```

then the entries in the file list would effectively have the following absolute path:

```
C:\website\files\www\index.html
C:\website\files\www\andre\index.html
C:\website\files\www\products.html
```

## File List Deployments from TeamSite Areas

In some cases, you might want a file list deployment to originate from a TeamSite area on the source host. Because the `filelist` attribute is only associated with the `sourceFilesystem` element, you cannot specify a TeamSite area. However, you can specify the file system path to that TeamSite area on the source host. For example, if you wanted to designate the following TeamSite area as the source area:

```
//IWSERVER/default/main/dev/EDITION
```

you would enter the file system location equivalent of the TeamSite area in the `sourceFilesystem` element's `area` attribute. For example:

```
area="/iwmnt/default/main/dev/EDITION/ed01012001" or
area="Y:\default\main\dev\EDITION\ed01012001"
```

Unlike TeamSite area-based areas, where you can simply state the value "EDITION" to indicate the latest TeamSite edition available, when entering a file system-based path to a TeamSite area, you must enter the entire path to the specific edition you want, even if it is the most recent one.

## Defining the Target Host Location

The same principles apply to the target host area for file list deployments as for directory or TeamSite comparison deployments. The target host area must be a file system location. However, unlike in a directory comparison deployment, the target host area in a file list deployment is not being compared to the files in the source host area. Rather, the target host area is simply a location where the files deployed as specified in the file list end up. See “Defining the Target Host Area” on page 213 for general configuration information, including specifying a location within the file system-based target area.

## Defining Source-Based Overrides

You can override global target-side attributes, such as the target file location and certain rules and filters with alternate attributes on a source-level basis. This feature allows files in a specified source file location to be deployed to an alternate target file location, and also overrides any existing target-side rules or filters. You can also use this feature to deploy a single set of files to multiple target file locations within the same deployment configuration.

One use of source-based overrides is when there are metadata files associated with a set of deployed files. You might want to also deploy those metadata files, but to a different target file location. By specifying this alternate target file location in conjunction with the metadata source file location, you can deploy both the main files and the metadata files within the same definition.

Source-based overrides are specified in the `targetRules` element. When this element is present in the deployment configuration, OpenDeploy will override the target file location in the `targetFilesystem` element and both compare (if necessary) and send the files to the path location specified as the `targetRules` element’s `area` attribute value instead.

The `targetRules` element resides within the `pathSpecification` element associated with each occurrence of the `sourceFilesystem` or `sourceTeamsite` elements in a deployment configuration. Each `sourceFilesystem` or `sourceTeamsite` element can have one occurrence of the `targetRules` element within each occurrence of the `pathSpecification` element.

The following example illustrates how a source file location can compare and deploy files to an alternate target file location (D:\metadata\files) at the same time another source file location deploys to the standard target file location (D:\website\files), all within the same definition:

```
<source>
  <sourceFilesystem area="C:\website\files">
    ...
  </sourceFilesystem>

  <sourceFilesystem area="C:\metadata\files">
    <pathSpecification>
      ...
      <targetRules area="D:\metadata\files" />
    </pathSpecification>
  </sourceFilesystem>
</source>

<target useReplicationFarm="MYFARMNAME">
  <targetFilesystem area="D:\website\files">
  </target>
```

You can specify source-based overrides for the following OpenDeploy features:

- Filters
- Transfer rules
- Comparison rules
- Permission rules

See Chapter 5, “Advanced Features” for more information on these features.

## Defining Target-Based Overrides

There are a variety of situations where the single set of values specified in the `target` element of a deployment configuration cannot apply to all the target hosts in a multi-target deployment equally or successfully. These situations include:

- The target hosts include both Windows and UNIX servers.
- A different target file location is required for one or more target hosts.
- Features need to be added or modified on a target-specific basis.

### Mixed Platform Target Areas

If all the target host servers in a multi-target deployment use the same path syntax, you can use the `area` attribute in the `targetFilesystem` element to specify the target host area. However, in cases where you have mixed platform targets, such as a combination of Windows and UNIX hosts, no single `area` attribute value can accurately specify all the target areas. For example, if you wanted to deploy to the directory `files` on your target hosts, the path on the target hosts might be the following:

Windows — `C:\files` *or*

UNIX — `/etc/files`

To adjust for mixed platform targets, you can override the general values found in the `targetFilesystem` elements with target-specific values in each target hosts' `nodeRef` element. Within each `nodeRef` element you have the option of specifying a `targetRules` element and its associated `area` attribute for that target host. The `targetRules` element's `area` attribute value specified in a target host's `nodeRef` element overrides the `targetFilesystem` element's `area` attribute value for both directory comparison purposes and for receiving the deployed files.

In deployment types where the target area is not involved in comparing files, such as TeamSite comparison or file list deployments, then the `area` attribute in the `nodeRef` element determines where the deployed files are located on the target host.

The absence of the `targetRules` element in the `nodeRef` element indicates that the target will accept the `area` attribute value specified in the `targetFilesystem` element. However, if this `area` attribute value's path syntax does not match that of the target host platform, then the deployment will fail for that target. In the following example, a deployment configuration has specified the following `area` attribute value within the `targetFilesystem` element:

```
<targetFilesystem area="C:\website\files" />
```

There are two target hosts: *mars*, a Windows server and *venus*, a UNIX server. The target host *mars* can accept this target host location, but *venus* cannot. Therefore, it is necessary to add an `area` attribute value to the `nodeRef` element for *venus*:

```
<replicationFarmSet>
  <replicationFarm name="MYFARMNAME">
    <nodeRef useNode="mars" />
    <nodeRef useNode="venus">
      <targetRules area="/etc/website/files"/>
    </nodeRef>
  </replicationFarm>
</replicationFarmSet>
```

Now when the deployment is run, *mars* can accept the target path of the following:

```
C:\website\files
```

as specified in the `targetFilesystem` element while *venus* can accept its own internal path the following:

```
/etc/website/files
```

and the deployment can be successful.

## Specifying Different Target Areas

You can also use this feature to specify different target host areas even when the source and target hosts are all of the same platform. In the following example, the target hosts *mars*, *jupiter*, and *saturn* are all Windows servers.

```
<replicationFarmSet>
  <replicationFarm name="MYFARMNAME">
    <nodeRef useNode="mars" />
    <nodeRef useNode="jupiter">
      <targetRules area="c:\website\western"/>
    </nodeRef>
    <nodeRef useNode="saturn">
      <targetRules area="c:/website/eastern"/>
    </nodeRef>
  </replicationFarm>
</replicationFarmSet>
```

Using the same `targetFilesystem` element and `area` attribute values from before, *mars* accepts that area location to receive the deployed files, while *jupiter* and *saturn* each override that location with ones unique to their own file systems.

## Defining Features on a Target-Specific Basis

You can specify certain features to apply to particular target hosts. A feature defined within the `nodeRef` element for a target host overrides any comparable feature or attribute value defined within the `target` element of the deployment configuration, as well as any source-based overrides specified within the `pathSpecification` element. Within each `nodeRef` element you can add additional elements to specify features for that target host including:

- Filters
- Transfer rules
- Comparison rules
- Permission rules

See Chapter 5, “Advanced Features” for information on these features.

## Deployment Logging

Logging settings in a deployment configuration affect the following types of log files:

- Source macro log
- Source micro log
- Receiver macro log
- Receiver micro log

Base server and receiver log files are not affected by logging settings present in deployment configurations.

You can specify the following logging rules in a deployment configuration:

- Rollover threshold size for a deployment's micro and macro log files
- Logging level

One or both of these settings will override any equivalent settings present in the base server or receiver configurations as they apply to deployment logs. Logging level settings in a deployment configuration can be overridden only when a different level is specified when a deployment is started manually in the OpenDeploy user interface, or by using the `iwodstart` command.

You cannot specify a different log file location in a deployment configuration. That functionality only exists in the base server or receiver configurations files.

See “Logging” on page 155 for a complete description of how OpenDeploy logs deployments, and on how to manage your log files.

## Transactional Deployments

*Transactional deployments* give you an added level of security for maintaining the integrity of your Web sites during deployments by automatically rolling back a deployment and restoring the target host files to their previous states in the event one or more target deployments fail. Transactional deployments are particularly useful where a number of recipient target hosts must have their Web sites synchronized with each other. If one or more of the deployments to these target hosts fail, it may be preferred to restore some or all of them (even the target hosts whose deployments succeeded) back to their previous states until another deployment can be attempted.

In Figure 45, the source host *mars* attempts a transactional deployment to the target host *venus*. The deployment can be of any type. During the deployment, file transmission to the target host is interrupted halfway through the process and is considered to be a failed deployment. After OpenDeploy becomes aware that this transactional deployment has failed, it restores *venus'* file area containing the deployed files back to its original state.

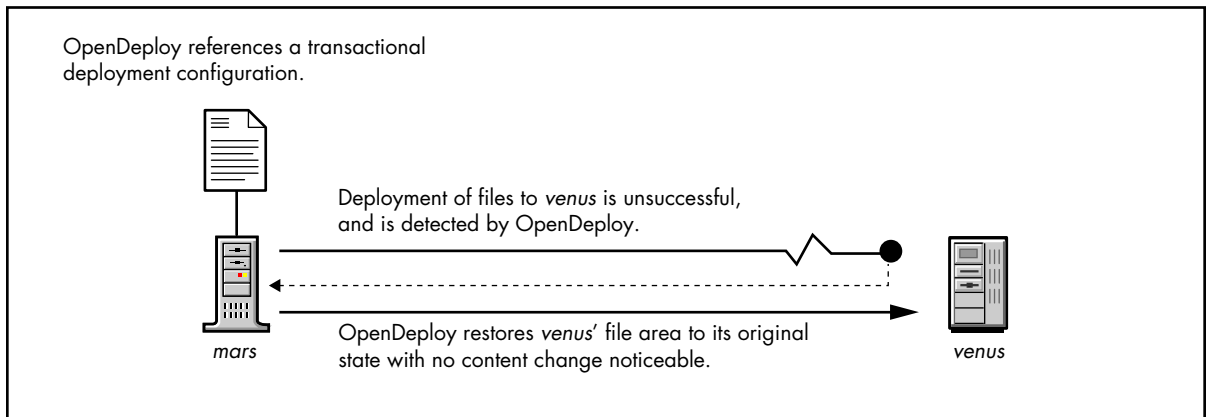


Figure 45: Transactional Deployment

Transactional deployments are enabled only in the deployment configuration file an attribute of the deployment element. Give the value `yes` if you want the deployment to be transactional. For example:

```
<deployment transactional="yes">
```

## Use with Multi-Tiered Deployments

Transactional deployments are configurable between a single source host and its target hosts. If you want to configure a multi-tiered deployment to be transactional, configure each tier to support transactional deployments at that level separately. The first tier source host in a multi-tiered deployment cannot automatically pass its transactional deployment configuration to subsequent tiers. See “Multi-Tiered Deployments” on page 233 for more information.

## Fan-Out Deployments

OpenDeploy can deploy the same set of files to multiple target hosts as part of a single deployment. Deploying to multiple targets in this way is called a *fan-out deployment*. Fan-out deployments are often preferred if your organization has several production Web servers, each of which must have its Web content synchronized with the others. A fan-out deployment automatically deploys to all target hosts simultaneously with no more effort on your part than if you were deploying to a single target host.

The same deployment configuration used to deploy files to a single target host can be modified to include all targets. Any type of deployment can be used, and all deployment rules and features are allowed in fan-out deployments. Although deployment to each target is identical by default, you can also modify the fan-out deployment configuration to allow exemptions to the rules on a target-specific basis.

In Figure 46, the source host *mars* performs a fan-out deployment to three mirrored production servers: *venus*, *jupiter*, and *mercury*.

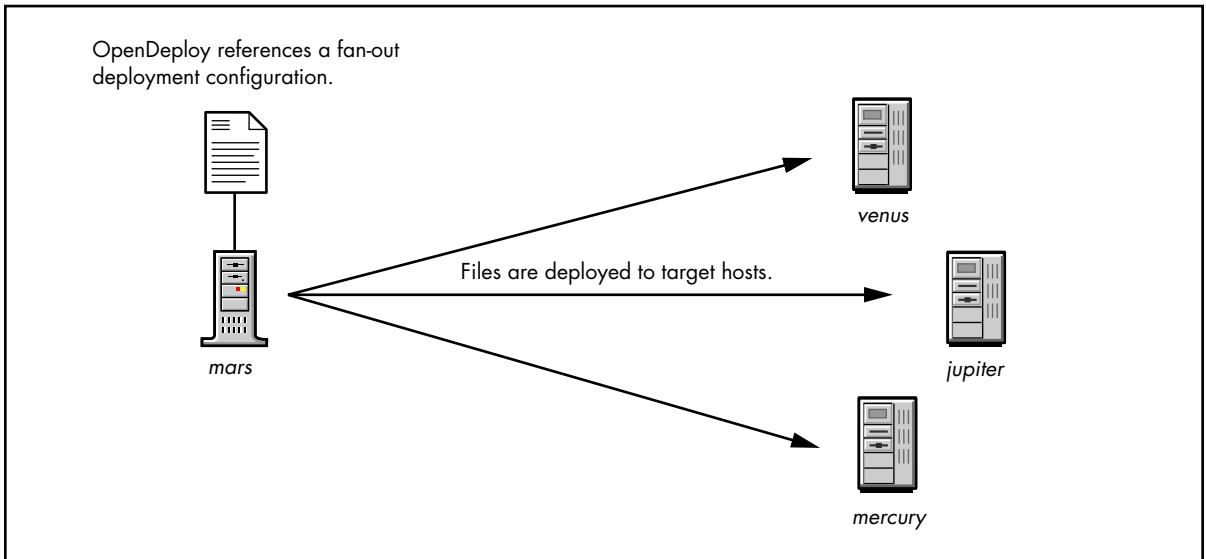


Figure 46: Fan-Out Deployment

You configure fan-out deployments using one of the following methods:

- Use multiple `nodeRef` elements within the `replicationFarm` element:

```
<replicationFarm>
  <nodeRef useNode="venus" />
  <nodeRef useNode="jupiter" />
</replicationFarm>
```

The `useNode` attribute value in a `nodeRef` is the logical name for a specific target host. That logical name must appear in the source host's nodes configuration file (by default `odnodes.xml`).

- Use multiple `execDeploymentTask` elements, each of which can reference a named definition element. That element in turn specifies a grouping of one or more source file locations with a single target file location:

```
<deployment ...>
  <execDefinitionTask useDefinition="MyFirstDef" />
  <execDefinitionTask useDefinition="MySecondDef" />
</deployment>
```

The `useDefinition` attribute value in an `execDefinitionTask` element references a particular definition elements in the configuration, and deploys files specified within that definition element when the deployment is run.

## Transactional Targets in Fan-Out Deployments

Because it is often required that all the target hosts in a fan-out deployment are mirror images of each other as well as the source host, it might be preferable not to deploy files to any of the target hosts in a fan-out deployment, unless a percentage of the targets, or one or more targets in particular, successfully receive the files. If any target does not receive its deployed files successfully, that deployment is considered to be a failure. You can restore those targets that did receive deployed files back to their previous existing states by enabling the transactional feature. See “Transactional Deployments” on page 229 for more information.

You can designate all target hosts to be transactional in a fan-out deployment by specifying a value of `yes` for the `transactional` attribute in the `deployment` element of the fan-out deployment configuration. For example:

```
<deployment transactional="yes">
```

In Figure 47, the source host *mars* performs a deployment-wide transactional fan-out deployment to two target hosts: *jupiter* and *venus*. Although the deployment to *jupiter* was successful, the deployment to *venus* failed. Because this fan-out was transactional on a deployment-wide basis, after the deployment was determined to be a failure, *mars* automatically restores both the target hosts.

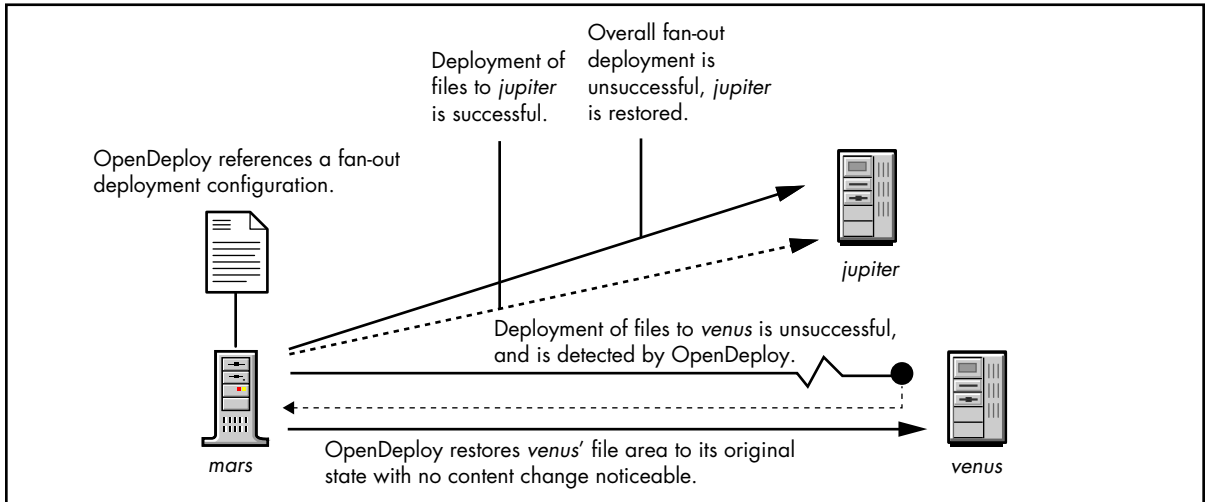


Figure 47: Transactional Fan-Out Deployment

## Multi-Tiered Deployments

A *multi-tiered deployment* is a single-target or fan-out deployment where one or more of the target hosts redeploy the files to a new group of target hosts. Each combination of source host and target hosts is known as a *tier*. Any host that redeploys received files must have the base server software installed to send files. The next-tier source host redeploying the files references its own deployment configuration files for the next-tier deployment.

The original deployment configuration information is not transmitted from the first source host to the next-tier source host as part of the deployment. Any combination of deployment types and OpenDeploy features can be utilized in a multi-tier deployment. There is also no limit to the number of tiers that can be included, as long as each tier has at least one host with the base server software installed.

In Figure 48, the source host *mars* deploys a fan-out deployment to its target hosts: *venus*, *jupiter*, and *mercury*. This represents the first tier. The target hosts *venus* and *mercury* have the receiver software installed, allowing them only to receive deployments. However, *jupiter* has the base server installed and can send as well as receive deployed files. After it successfully receives the files deployed from *mars*, *jupiter* references its own deployment configuration file and redeploys the files to its own target hosts: *saturn* and *pluto*. The source host *jupiter* and its targets represent the second, or next, tier.

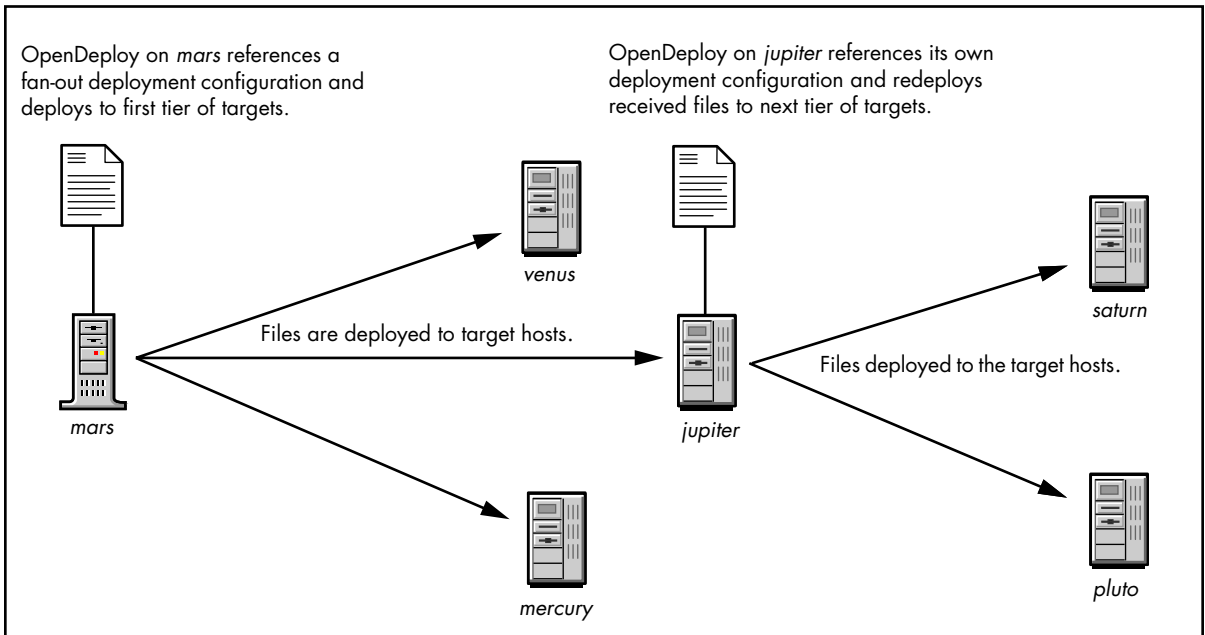


Figure 48: Multi-Tiered Deployment

Any target hosts with the base server software installed can start their own deployments, passing on the same deployment data to new targets. This process of redeploying files can continue indefinitely if every tier has a source host capable of redeploying the files it receives.

The role of the source host originating the deployment stops after its target hosts successfully receive the deployed files. The source host at the second tier now takes over, and its deployment configuration file determines the scope and functionality of the deployments on this tier. The second-tier source host also does not receive any deployment configuration information from the first-tier source host.

An OpenDeploy server that is intended to participate in multi-tiered deployments as a secondary- or later-tiered source host must have its deployment configuration file properly configured before the deployment.

The ability of an OpenDeploy sending host to redeploy files it receives is defined in the `nextDeployment` element of a deployment configuration file.

The `nextDeployment` element has the following attributes:

- `deployment` — enter the name of the deployment that will execute on the target host upon completion of this current deployment. The deployment name will be the same as the deployment configuration file.
- `invokeOnSuccess` — (yes|no) indicate whether or not the source host must receive a successful deployment confirmation before deploying its files. If the value is `yes`, then the OpenDeploy server will only start its own deployment to its tier when it receives word that the overall fan-out deployment was successful.

In the following example, a next-tier source host in a multi-tiered deployment has the following entry in the deployment configuration file:

```
<nextDeployment deployment="monthly" invokeOnSuccess="yes" />
```

This configuration indicates the following:

- The configuration that will be used is `monthly`, and that the configuration for `monthly` (`monthly.xml`) resides in the `od-home/conf` directory of the node that will redeploy the files.
- The deployment can only occur if the OpenDeploy host received confirmation from the previous OpenDeploy source host that the overall fan-out deployment was successful.

## Reverse Deployments

A *reverse deployment* allows new or updated files residing on what is typically a target host (often a production server) to be deployed back to the source host (often a development server) that normally sends deployments. In this relationship, the *reverse host* deploys files to the *reverse target*. Unlike a typical forward deployment, the reverse source can be a host with only the receiver software installed. The reverse target (the host with the base server software installed) manages the reverse deployment, and the reverse source must be listed as a target node in the nodes configuration file of the reverse target host. Reverse deployment files reside in the same directory on the host as other deployments.

Reverse deployments are often used to deploy files generated on production servers back to the development server. For example:

- Web server log files
- Data files created via a CGI application
- Assets uploaded through a Web server application

In Figure 49, the production server *venus* has generated files that need to be reverse deployed back to the development server *mars*. As the reverse target, *mars* initiates a reverse deployment from the reverse source host *venus*. The deployment then takes place like any other deployment.

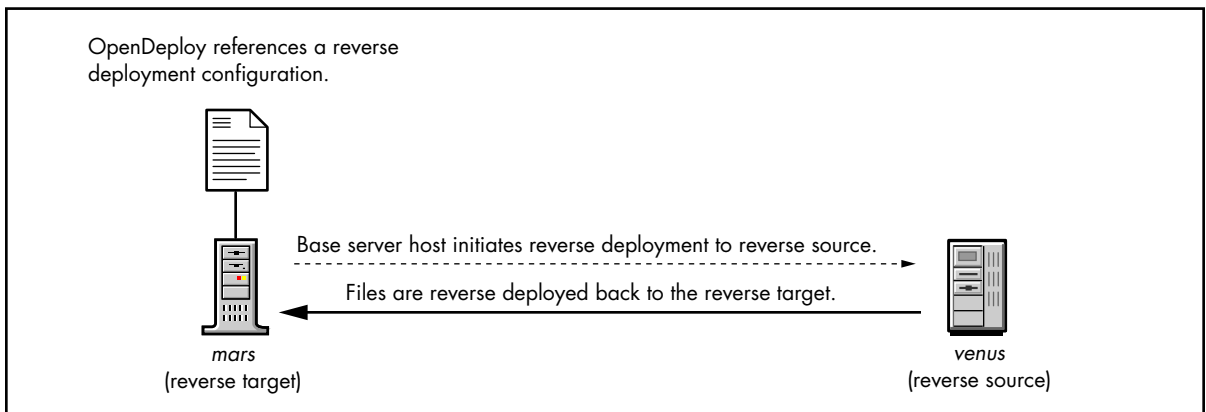


Figure 49: Reverse Deployment

Reverse deployments are configured in a manner similar to traditional deployments, except the source and target host roles are switched as the reverse source and reverse target. The `reverseSource` and `reverseTarget` elements accommodate this change in host roles.

The `reverseSource` element identifies the attributes regarding the sender of the deployment during a reverse deployment. The `useReplicationFarm` attribute specifies the target replication farm to which the reverse source host belongs.

The `reverseTarget` element identifies the attributes regarding the recipient of the deployment during a reverse deployment.

```
<definition name="reverse_deployment">
  <reverseSource useReplicationFarm="MYFARMNAME">
    <sourceFilesystem area="C:\dev\website\files">
      <pathSpecification>
        <path name="monthly" />
      <pathSpecification>
    </sourceFilesystem>
  </reverseSource>
  <reverseTarget>
    <targetFilesystem area="D:\website\files" />
  </reverseTarget>
</definition>
```

## Using Sample Configurations

OpenDeploy provides a variety of configuration samples. You can view the syntax and comments in these sample configurations to better understand how to modify a deployment configuration file. You can also modify sample configurations to work within your own OpenDeploy environment, relieving you from having to compose deployment configurations in their entirety. These files are located on your OpenDeploy server at the following location:

*od-home/conf/examples*

In this location you will find the following sample configurations:

- `deploysimple.xml` — a fan-out deployment configuration with a small number of additional features. It includes numerous comments to assist you to understand each portion of the file.
- `deployadvanced.xml` — a fan-out deployment including many of the more advanced OpenDeploy features, including filtering, various rules, and Deploy and Run. It includes numerous comments to assist you to understand each portion of the file.
- `single.xml` — a single-target deployment configuration. It does not contain comments, but does indicate where you can substitute your own system and configuration information for the variables it contains. This allows you to simply modify the existing file for your own use, rather than having to create a completely new deployment configuration.
- `fanout.xml` — a fan-out deployment configuration. It does not contain comments, but you can modify it for your own use.
- `multi.xml` — a multi-tiered fan-out deployment configuration. It does not contain comments, but you can modify it for your own use.
- `reverseDeploy.xml` — a reverse deployment configuration. It does not contain comments, but you can modify it for your own use.

You can use the sample files to perform deployments with only a minimum of modification. Because XML-based deployment configuration files must be syntactically correct to work, writing new deployment configuration files can result in a large amount of troubleshooting. By knowing which parts of a sample deployment configuration file to modify for your use and which parts to leave alone, you can become productive more quickly and avoid many problems.

The sample configuration for a simple deployment is `single.xml`, which will deploy files from a source host to a single target host. This is a good sample file from which to learn how to deploy files. Here is the source code for `single.xml`:

```
<?xml version="1.0" encoding="UTF-8"?>

<deploymentConfiguration>

    <localNode host="MYALLOWSENDINGHOSTNAME" />

    <replicationFarmSet>
        <replicationFarm name="MYFARMNAME">
            <nodeRef useNode="MyLocalHost" />
        </replicationFarm>
    </replicationFarmSet>

    <definition name="MYDEFINITIONNAME">
        <source>
            <sourceFilesystem
                area = "/MYOPENDEPLOY/conf/dtd">
                <pathSpecification>
                    <path name = "." />
                </pathSpecification>
            </sourceFilesystem>
        </source>
        <target useReplicationFarm="MYFARMNAME">
            <comparisonRules dateDifferent="yes" />
            <permissionRules file="0644"
                directory="0755" />
        </target>
    </definition>

    <deployment transactional="no">
        <execDeploymentTask useDefinition="MYDEFINITIONNAME" />
    </deployment>

    <logRules maxBytes="32Mb" level="verbose" />

</deploymentConfiguration>
```

You can use this sample file to deploy from a directory on your source host to any other host in your enterprise that has the OpenDeploy base server or receiver software installed. To modify and use this sample configuration to deploy files from your source host, perform the following:

## List Your Target Hosts in the Nodes Configuration File

Ensure that your source host's node configuration file contains a `node` element entry for your target host. This entry must include the following information:

- The target host's fully qualified DNS host name or IP address.
- The target host's "listener" port. The default value during installation is 20014.
- A logical name that you designate for the target host. You will use this logical name in the deployment configuration files to indicate the target of the deployment.

See "Defining Target Host Nodes" on page 70 for more information.

## Modify the Target Host to Accept Your Deployment

Ensure that your target host's base server or receiver configuration file permits the receiving of deployments from your source host. You must list your source host as a `node` element entry within the `allowedHosts` element. This is similar to the adding of your target host to your node configuration file in the previous task. See "Specifying Allowed Hosts for Received Deployments" on page 83 for more information.

You will also need to specify one or more directory locations on the target host that can receive deployed files from your source host. See "Specifying Allowed Directories for Deployments" on page 84.

## Specify Your Source Host's Name

Add the fully qualified host DNS host name or IP address of your source host in the `localNode` element:

```
<localNode host="MYALLOWSENDINGHOSTNAME" />
```

For example, if your host's IP address was 114.342.23.21, then the `localNode` element would be:

```
<localNode host="114.342.23.21" />
```

## Specify the Target Host's Logical Name

Add the `nodeRef` element's `useNode` attribute value to reflect the logical name of your target host.

Each deployment configuration contains a `replicationFarmSet` element and its child elements.

```
<replicationFarmSet>
  <replicationFarm name="MYFARMNAME">
    <nodeRef useNode="MyLocalHost" />
  </replicationFarm>
</replicationFarmSet>
```

You can give the `replicationFarm` element a unique name value for its `name` attribute, but here it is not necessary. The value can remain `MYFARMNAME`.

Each target host is listed within the `replicationFarm` element as a separate `nodeRef` element. Since this is a single target deployment, you only need to include that target's logical name as the value for the `useNode` attribute. If your target host has a logical name of *venus* specified in your server's nodes configuration file, then use that logical name here:

```
<nodeRef useNode="venus" />
```

## Specify the Deployment Type

The `definition` element and its child elements indicate the following information:

- Information about the source of the deployment.
- Information about the targets of the deployment.

You can give the `definition` element a unique value for its `name` attribute, but here it is not necessary. The value can remain `MYDEFINITIONNAME`.

Within the `definition` element is the `source` element.

```
<definition name="MYDEFINITIONNAME">
  <source>
    <sourceFilesystem area="/MYOPENDEPLOY/conf/dtd">
      <pathSpecification>
        <path name = "." />
      </pathSpecification>
    </sourceFilesystem>
  </source>
  <target useReplicationFarm="MYFARMNAME">
    <comparisonRules dateDifferent="yes" />
    <permissionRules file="0644" directory="0755" />
    <targetFilesystem area="/tmp" />
  </target>
</definition>
```

The `source` element contains all the elements and attributes that indicate the following:

- Type of deployment that is occurring.
- Directory or TeamSite area where the source files are located.
- Special features such as modifications to the deployment criteria.

Our example is of a directory comparison deployment. A directory comparison deployment will compare the files residing in a specified location on the source host with the files residing in a specified location on the target host. Those files that meet the deployment criteria based on differences in size, modification date, ownership (UNIX only), and other conditions, are moved to the specified target host location. A directory comparison deployment is indicated by the presence of the `sourceFilesystem` and `targetFilesystem` element in the configuration. See “Directory Comparison Deployments” on page 211 for more information.

## Specifying the Source Host File Locations

Within the `sourceFilesystem` element is the `area` attribute. Here is where the location of files on the source host are specified.

```
<source>
  <sourceFilesystem area="/MYOPENDEPLOY/conf/dtd">
    <pathSpecification>
      <path name = "." />
    </pathSpecification>
  </sourceFilesystem>
</source>
```

The `area` attribute’s value is the full path to the directory containing the files that will participate in the deployment. For example:

```
area="C:\reports" or
area="/etc/reports"
```

You can specify a subdirectory within the source file location using the `path` element and its `name` attribute. This value is a subdirectory or a path to a subdirectory relative to the `area` location. However, in this example no further location within the source `area` is required, so the `path` element’s `name` attribute value is “.”

## Specify the Target Host File Location

You must specify the location on the target host where the deployed files will reside following the deployment.

```
<target useReplicationFarm="MYFARMNAME">
  <comparisonRules dateDifferent="yes" />
  <permissionRules file="0644" directory="0755">
  </permissionRules>
  <targetFilesystem area="/tmp" />
</target>
```

The `target` element and its child elements and attributes include the location where the deployed files will reside on the target host after the deployment, and any optional features you want to augment the deployment criteria. The `target` element's `useReplicationFarm` attribute value must match the `replicationFarm` element's `name` attribute value you specified earlier. In this example you can leave it as `MYFARMNAME`.

In this example, the `permissionRules` element specifies the permissions to be set on deployed files and directories on a UNIX system. However, that is a more advanced topic, so it will not be covered here.

The target location is specified in the `targetFilesystem` element's `area` attribute. In other types of deployments, you can specify the target location as a TeamSite area instead. The `targetFilesystem` element's `area` attribute value is the full file system path to where the deployed files will reside on the target host. Enter the path to the directory where you want the deployed files to reside, for example:

```
area="C:\reports\western" or
area="/etc/reports/western"
```

## Specify Transactional

In the final section of our example deployment is the `deployment` element and its child elements and attributes.

```
<deployment "transactional="no">
  <execDeploymentTask useDefinition="MYDEFINITIONNAME" />
</deployment>
```

The `transactional` feature will automatically cause OpenDeploy to roll back a deployment and restore the target host files to their previous state in the event one or more target deployments are unsuccessful. However, for this example deployment, the feature is not needed. You can leave the default value of `no` as is.

## Modifying the Sample Configuration to Fit Your Needs

Now that you have covered the important parts of the sample deployment configuration, you can modify its use in your own enterprise. In the next example, the original deployment configuration has been modified to reflect the following setup of an actual OpenDeploy source host using the following values. The element and attribute containing the changed value are noted in parentheses:

- Host name of the OpenDeploy server sending the deployment (`localNode host`):

```
114.342.23.21
```

- Logical name of target host (`nodeRef useNode`):

```
venus
```

- Location on source host where files reside (`sourceFilesystem area`)

```
C:\reports
```

- Location on target host where deployed files will reside (`targetFilesystem area`)

```
C:\reports\western
```



In the following code example, the changed values are in bold. All other elements and attributes can be used without modification.

```
<?xml version="1.0" encoding="UTF-8"?>

<deploymentConfiguration>

  <localNode host="114.342.23.21" />

  <replicationFarmSet>
    <replicationFarm name="MYFARMNAME">
      <nodeRef useNode="venus" />
    </replicationFarm>
  </replicationFarmSet>

  <definition name="MYDEFINITIONNAME">
    <source>
      <sourceFilesystem area = "C:\reports">
        <pathSpecification>
          <path name = "." />
        </pathSpecification>
      </sourceFilesystem>
    </source>
    <target useReplicationFarm="MYFARMNAME">
      <comparisonRules dateDifferent="yes" />
      <permissionRules file="0644" directory="0755">
      </permissionRules>
      <targetFilesystem area="C:\reports\western" />
    </target>
  </definition>

  <deployment transactional="no">
    <execDeploymentTask useDefinition="MYDEFINITIONNAME" />
  </deployment>

</deploymentConfiguration>
```

To use this example, you must also modify the base server or receiver configuration file of the host receiving the deployment (depending on whether that recipient host has the base server or receiver software installed).

After you update the recipients host's server configuration, you must reset it either by restarting OpenDeploy, or by using the `iwodserverreset` command-line tool. See "Starting OpenDeploy" on page 109 and "Refreshing the OpenDeploy Server" on page 115 for more information.

After you have finished modifying the sample deployment configuration file to function within your OpenDeploy environment, place some files in the source area directory and try running a deployment. See "Starting a Deployment" on page 141 for more information.

After you have become comfortable with how to modify a sample file to work within your OpenDeploy environment, you can make more modifications to the `single.xml` configuration file used in the example, or try some of the other example files. You can add additional target hosts to the nodes configuration file and the `nodeRef` element and turn your single deployment into a multi-target fan-out deployment. You can also modify the `source` and `target` elements, and change the deployment type from a directory comparison to a TeamSite comparison or file list deployment. Finally, you can also add additional features such as filters or rules described later in this manual.



## Chapter 5

# Advanced Features

---

This chapter describes advanced OpenDeploy deployment features and how they are configured.

## Filtered Deployments

You can modify the deployment configuration to filter which directories are included in a deployment. These exclusions can be based on one or both of the following criteria:

- File system location (`excludePath` element)
- File name pattern (`excludePattern` element)

Both types of filtered deployment elements reside within the `filters` element. You can have multiple instances of either element in any combination within the deployment configuration.

## File System Location-Based Exclusions

File system location-based deployment exclusions ignore files located in a specified path during the deployment. You must modify the deployment configuration file's `excludePath` element and its `subPath` attribute to specify the source and target host path locations of the files to be excluded. The path specified in the `subPath` attribute is relative to the local directory or TeamSite area on the host containing the files to be moved. That local directory or TeamSite area is specified in the `area` attribute value of the `sourceFilesystem` or `sourceTeamsite` elements, respectively. The example below shows the `excludePath` element and `subPath` attribute as they appear in the deployment configuration file:

```
<pathSpecification>
  ...
  <filters>
    <excludePath subPath="path_to_be_excluded" />
  </filters>
</pathSpecification>
```



For example, if you wanted to exclude the directory `monthly`, which resides within the specified area, then the element would be:

```
<excludePath subPath="monthly" />
```

You can specify multiple occurrences of the `excludePath` element in your deployment configuration file by adding an additional occurrence of the `excludePath` element for each exclusion. For example:

```
<filters>
  <excludePath subPath="monthly" />
  <excludePath subPath="quarterly" />
</filters>
```

Filtered deployments are defined in the `filters` element of the configuration in the following manner. The specific types of filtering, and where they are located within the configuration file, are determined by the elements and attributes specified within the `filters` element.

```
<source>
  <sourceFilesystem area="C:\dev\website\files">
    <pathSpecification>
      <path name="." />
      <filters>
        <excludePath subPath="path" />
        <excludePattern regex="pattern_expression" />
      </filters>
    </pathSpecification>
  </sourceFilesystem>
</source>
```

where *path* is the path to be excluded from the deployment, and the *pattern\_expression* is a regular expression for ignoring specified directories during the deployment.

## Pattern-Based Exclusions

Pattern-based exclusions ignore directories based on a specified name-exclusion criteria on the source and target hosts during deployments. You must modify the deployment configuration file's `excludePattern` element and its `regex` attribute to specify the name patterns of the directories to be excluded. For example:

```
<filters>
  <excludePattern regex="pattern_for_exclusion" />
</filters>
```

The `regex` attribute value must be a regular expression pattern. The directories compared with the exclusion pattern are those relative to the local directory or TeamSite area on the target host. For example, if you wanted to exclude any path relative to the target host area with the name `internal`, the `excludePattern` element value would be:

```
<excludePattern regex="internal" />
```

You can specify multiple exclusionary patterns for a deployment by adding another occurrence of the `excludePattern` element for each excluded pattern. For example:

```
<excludePattern regex="internal" />
<excludePattern regex="external" />
```

or use wildcard terms to indicate multiple patterns, for example:

```
<excludePattern regex="??ternal"/>
```

If you are not familiar with regular expressions, consult a reference manual such as *Mastering Regular Expressions* by Jeffrey Friedl.

## Configurations

These filter exclusions can be present on both the source and target hosts. Filesystem location-based exclusions can reside in different parts of the deployment configuration depending on the type:

- Source-side exclusions
- Target-side exclusions



## Source-Side Exclusions

In source-side exclusions, the `filters` element and its `excludePath` and `excludePattern` child elements can appear both within the `pathSpecification` and the `targetRules` elements. For example:

```
<source>
  <sourceFilesystem area="C:\dev\website\files">
    <pathSpecification>
      <path name="." />
      <filters>
        <excludePath subPath="path" />
        <excludePattern regex="pattern_expression" />
      </filters>
      <targetRules>
        <filters>
          <excludePath subPath="path" />
          <excludePattern regex="pattern_expression" />
        </filters>
      </targetRules>
    </pathSpecification>
  </sourceFilesystem>
</source>
```

## Target-Side Exclusions

In target-side exclusions, the `filters` element and its `excludePath` and `excludePattern` child elements can appear both within the `target` element:

```
<target>
  <targetFilesystem area="D:\website\files" />
  <filters>
    <excludePath subPath="path" />
    <excludePattern regex="pattern_expression" />
  </filters>
</target>
```

and within a `targetRules` element at the node level:

```
<replicationFarmSet>
  <replicationFarm name="MYREPLICATIONFARM">
    <nodeRef useNode="venus">
      <targetRules>
        <filters>
          <excludePath subPath="path" />
          <excludePattern regex="pattern_expression" />
        </filters>
      </targetRules>
    </nodeRef>
  </replicationFarm>
</replicationFarmSet>
```

### Override Precedence

Filters specified for the `targetRules` element within the `pathSpecification` element override any filters specified for the `target` element.

Filters specified for the `targetRules` element within the `nodeRef` element override any filters specified for the `targetRules` element within the `pathSpecification` element and any filters specified for the `target` element.

## Target Host File Deletions Using Filtered Deployments

A file or directory that is excluded from a filtered deployment is treated as being non-existent at the source host. If the `doDeletes` attribute of the `transferRules` element has a value of `yes`, then the same file at the target host will be deleted during the deployment. This is because the `doDeletes` attribute removes any target host file that does not have an equivalent file at the source host. Filtered deployments can inadvertently cause target-side files to be deleted in this manner.

Filters specified on the target side cause those files to be ignored during comparisons and thus files with the same name on the source side will always be deployed since they will always appear to be missing on the target side. Be careful if you are using the `doDeletes` feature in conjunction with filtered deployments. See “File Transfer Rules” on page 256 for more information.

## File Comparison Rules

The default comparison criteria used to determine whether or not a given file should be deployed are:

- Modification date (when the source-side file is newer than its target-side equivalent)
- Type mismatch (a file and a directory sharing the same name)
- Size difference

In addition, OpenDeploy provides a variety of other deployment criteria you can use or ignore in your deployment configurations, including:

- Source-side file is older than target-side equivalent
- Access control list (ACL) difference (Windows only)
- User difference (UNIX only)
- Group difference (UNIX only)
- Permission difference (UNIX only)

You can customize some these criteria within a deployment configuration by setting various attributes within the `comparisonRules` element. Here is a listing of those attributes:

- `dateDifferent` — (yes|no) indicate whether or not a file should be deployed if there is any difference in file date (older or newer) between the source and target versions. This differs from the OpenDeploy default date-based comparison setting, where a file is deployed only if the source file is newer than the target file. A value of `yes` indicates that the file should be deployed. Default value is `no`. The `dateDifferent` attribute is mutually exclusive with the `revert` attribute.
- `revert` — (yes|no) indicate whether or not a file should be deployed if the source version is older than the target version. A value of `yes` indicates that the file should be deployed. Default value is `no`. The `revert` attribute is mutually exclusive with the `dateDifferent` attribute.
- `ignoreAcls` (Windows only) — (yes|no) indicate whether or not to ignore differences in the ACLs during the file comparison. Default value is `no`.
- `ignoreModes` (UNIX only) — (yes|no) indicate whether or not to ignore differences in the UNIX-based permission bit mask during the file comparison. Default value is `no`.

- `ignoreUser` (UNIX only) — (yes|no) indicate whether or not to ignore differences in the UNIX-based file user ownership during the file comparison. Default value is `no`.
- `ignoreGroup` (UNIX only) — (yes|no) indicate whether or not to ignore differences in the UNIX-based file group ownership during the file comparison. Default value is `no`.

You can enable and disable comparison rules in any combination. For example, in the following occurrence of the `comparisonRules` element:

```
<comparisonRules dateDifferent="yes" ignoreAcls="yes" />
```

OpenDeploy applies the following rules:

- A file will be considered for deployment if the source file modification date is either older or newer than the target file.
- Differences in access control list (ACL) settings between the source are ignored during the comparison.

Otherwise, all other comparison criteria are in effect.

Access options specific to UNIX are ignored when deploying to a Windows target host and access options specific to Windows are ignored when deploying to a UNIX target host. Therefore, you will not receive any errors if you define both:

```
ignoreAcls="yes" and  
ignoreUser="yes"
```

## File Transfer Rules

You can modify your deployment configuration to follow or ignore various file transfer-related rules through the `transferRules` element and its various attributes. Here is a listing of those attributes:

- `doDeletes` — (yes|no) indicate whether or not files and directories that reside in the target host area but not in the source host area should be deleted following the deployment. By default they are not. Default value is `no`.

This feature sometimes can cause inadvertent file deletions when used in conjunction with target-side filters. See “Target Host File Deletions Using Filtered Deployments” on page 253 for more information.

- `dontDo` — (yes|no) indicate whether or not to proceed with the deployment following the comparison. If the value is `yes`, the deployment will not occur. Default value is `no`.

Performing a deployment using this feature will log all simulated deployed files to the deployment log. This is a good tool to use to check and compare files without actually performing a deployment. Files that are logged as being deployed indicate a difference between what is on the source server and the target server. You can also enable this feature using the `iwodstart -sim` command-line tool, or the simulated deployment feature in the OpenDeploy user interface. See “Performing a Simulated Deployment” on page 145 for more information on the benefits of simulated deployments.

- `preserveAcls` (Windows only) — (yes|no) indicate whether or not to preserve the Windows access control lists (ACLs) when the files are moved. By default, OpenDeploy applies ACLs based on the ACLs already existing on the containing folders on the target host receiving the deployed files. Default value is `no`. See “Using OpenDeploy with ACLs” on page 262 for more information.
- `followLinks` (UNIX only) — (yes|no) indicate whether or not symbolic links on the source and target hosts will be followed when the files are moved. Default value is `no`.

Enabling the `followLinks` attribute within the `transferRules` element only affects the target side. If you want to apply this feature to the source side as well, you must enable it in the `sourceTransferRules` element. See “Deploying Symbolic Links” on page 264 for more information.

- `svrTryCount` (Windows only) — enter the number of times OpenDeploy will attempt to deploy the file to the target host. This feature works in conjunction with Microsoft IIS, and is designed to accommodate times of heavy production server traffic.
- `svrTryInterval` (Windows only) — enter the amount of time in seconds OpenDeploy waits between deployment attempts. This feature works in conjunction with Microsoft IIS, and is designed to accommodate times of heavy production server traffic.
- `svrTryDisableOverwrite` (Windows only) — (yes|no) indicate whether or not to disable the ability of OpenDeploy to deploy files to a server even if the `svrTryCount` and `svrTryInterval` elements are specified. This feature works in conjunction with Microsoft IIS, and is designed to accommodate times of heavy production server traffic. Default value is no.
- `rmReadOnly` (Windows only) — (yes|no) indicate whether or not you want a deployed file to be able to overwrite its read-only target equivalent. If this feature is enabled with a value of yes, OpenDeploy will remove the read-only attribute from the target file, allowing the deployment to occur. A value of no will prevent the overwriting. Default value is no.

You can enable and disable comparison rules in any combination. For example, in the following occurrence of the `transferRules` element:

```
<transferRules doDeletes="yes" preserveAcls="yes" />
```

OpenDeploy applies the following rules:

- Any files existing in the target host area but not in the corresponding source area will be deleted following the deployment.
- (Windows only) Access control list (ACL) information will be retained following the deployment.
- Access options specific to UNIX are ignored when deploying to a Windows target host and access options specific to Windows are ignored when deploying to a UNIX target host.

## File Permission Rules

You can modify your deployment configuration to follow or ignore various file permission-related rules through the `permissionRules` element and its various attributes. Here is a listing of those attributes:

- `amask` (UNIX only) — enter the bit mask (in octal) to be ANDed with the permission bits of all files and directories. The `amask` octal value combines with the existing permission bit value of the affected file. If a file has the existing permission value of 664 (-rw-rw-r--) and the `amask` attribute as the following value:

```
amask="770"
```

then the resulting permission for that file (664 AND 770) following the deployment would be 660 (-rw-rw----).

- `omask` (UNIX only) — enter the bit mask (in octal) to be ORed with the permission bits of all files and directories. The `omask` octal value combines with the existing permission bit value of the affected file. If a file has the existing permission value of 666 (-rw-rw-rw-) and the `omask` attribute as the following value:

```
omask="022"
```

then the resulting permission for that file (666 OR 022) following the deployment would be 644 (-rw-r--r--).

- `directory` (UNIX only) — enter the permissions (in octal) given to all deployed directories. For example, if you wanted deployed directories to have the permission “drwxrwx---”, then the resulting value would be:

```
directory="770"
```

- `file` (UNIX only) — enter the permissions (in octal) given to all deployed files. For example, if you wanted deployed files to have the permission “-rw-rw-r-x”, then the resulting value would be:

```
file="665"
```

- **group** (UNIX only) — enter the group assigned to all deployed files and directories. This attribute value must be a valid group name or group ID. For example:

```
group="tech_pubs" or
```

```
group="200"
```

You must also specify the **user** attribute if you use employ the **group** attribute.

- **user** (UNIX only) — enter the user who will own all deployed files and directories. This attribute value must be a valid user name or user ID. For example:

```
user="jdoe" or
```

```
user="105"
```

You must also specify the **group** attribute if you use employ the **user** attribute.

- **changeAccess** (Windows only) — enter a value that modifies the access control lists (ACLs) so that specified users have the designated rights. The new access control entry (ACE) for each specified user allows only the specified rights, discarding any existing ACE. In the following example:

```
changeAccess="{ jdoe:W, tech_pubs:NONE }"
```

any existing ACEs for **jdoe** and **tech\_pubs** are removed, **jdoe** is granted write access, and the group **tech\_pubs** has no access at all. Any other access rights that may have existed for other users are left unchanged. See “Using OpenDeploy with ACLs” on page 262 for more information.

- **setAccess** (Windows only) — enter a value that replaces the ACLs for the deployed files and directories. In the following example:

```
setAccess="{ jdoe:ALL, tech_pubs:RX }"
```

the existing ACL is removed and the user **jdoe** is granted full access. The group **tech\_pubs** has read access to the specified files. Any other access rights that may have existed for the file are removed. See “Using OpenDeploy with ACLs” on page 262 for more information.

Access options specific to UNIX are ignored when deploying to a Windows target host and access options specific to Windows are ignored when deploying to a UNIX target host.

In the following example:

```
<permissionRules
  directory="770"
  file="664"
  group="marketing"
  user="rroe"
/>
```

OpenDeploy applies the following rules:

- The deployed directories will have “drwxrwx---” access as indicated by the value 770.
- The deployed files will have “-rw-rw-r--” access as indicated by the value 664.
- All deployed directories and files are assigned to the group `marketing`.
- All deployed directories and files are owned by the user `rroe`.

## User and Group Ownership Transferral

You can switch UNIX-based user and group ownership of deployed files and directories as an option of the file permission rules. For example, upon deployment you might want to change the ownership of a set of files currently owned by the user *jdoe* to the user *rroe*. Similarly, you might want to change the group ownership from *tech\_pubs* on the source host to the group *marketing* at the target host.

This feature is defined for user and groups by the `userTranslation` and `groupTranslation` elements, respectively. Both of these elements are child elements of the `permissionRules` element. The `permissionRules` element must first be specified before using these two elements. Each element has the following attributes:

- `from` — enter the existing source user or group ID (or the numerical uid or gid value assigned to a particular user or group account on the UNIX source host). For example:

```
from="jdoe" or
from="105"
```

- `to` — enter the new target user or group ID. For example:

```
to="rroe" or
```

```
to="110"
```

You must determine the appropriate user and group IDs at both the source and target hosts before you modify these attributes. You can determine these by using the `id` command at the prompt on a UNIX host. Your server will respond by displaying your user ID (UID) and group ID (GID). Enter the following command on your UNIX prompt for more details regarding the usage of this command:

```
man id
```

In the following example:

```
<permissionRules>  
  <userTranslation from="jdoe" to="rroe" />  
  <groupTranslation from="100" to="200" />  
</permissionRules>
```

OpenDeploy will review the files in a deployment for those owned by user ID `jdoe` or group ID `100`, and assign those files a new user ownership of `rroe` and group ownership of `200` after they are deployed to the target host.

## Using OpenDeploy with ACLs

Access Control Lists (ACLs) on Windows servers have the following syntax:

`{ name:ACE, name:ACE, ... }`

where *name* is the ACL name and *ACE* is the Access Control Entry (ACE) type.

### ACL Names

The ACL name can be one of the following:

- User name
- Group name
- Domain name\user name
- Domain name\group name

### ACE Types

ACEs consist of either of the following:

- Perm bits
- Standard perms

#### Perm Bits

*Perm bits* are sequences made up of one or more of the following characters:

- R — read
- W — write
- X — execute
- D — delete
- P — change permissions

- O — take ownership; equivalent to RWX

### Standard Perms

*Standard perms* consists of one of the following:

- ALL — RWXDPO
- NONE — none
- READ — RX
- WRITE — W
- CHANGE — RWXD

In the following example:

```
setAccess={ andre:ALL, everyone:RX }
```

OpenDeploy would remove the existing ACL and grant the user *andre* full access and the group *everyone* read access to the specified files.

In the following example:

```
changeAccess={ chris:ALL, everyone:RX }
```

would remove any existing ACEs for *chris* and *everyone*, and grant *chris* full access and the group *everyone* read access to the specified files. Any other existing ACEs would remain unchanged.

## Deploying Symbolic Links

By default, a symbolic link will be transferred intact and will point to the same relative or absolute path on the target side that it was pointing to on the source side. OpenDeploy will not deploy the actual file itself, nor will it validate the link on the target side to ensure the pointed-to files reside on the target host. However, you can elect to have OpenDeploy deploy the actual pointed-to files by enabling the follow links feature. This feature is not available with OpenDeploy running on Windows.

In the following example, *foo* is a link that points to the file `/etc/reports.txt`. If you enter the following command at the prompt:

```
ls -l foo
```

the return would be

```
foo -> /etc/reports.txt
```

If *foo* is moved as part of a deployment with the follow links feature enabled at the source host side, the deployment would find the file `/etc/reports.txt` and deploy it to the target as a new version of *foo*, replacing the one already there. This feature is useful if the source host area contains links, but the files they point to reside outside of the area and would otherwise not be included in the deployment.

### Source-Side

If you want to move the items pointed to by links contained on the source host area, you can enable the follow links feature with the `followLinks` attribute of the `sourceTransferRules` element. For example:

```
<sourceTransferRules followLinks="yes" />
```

### Target-Side

On the target side, if you want to replace whatever the link is pointing to with what was deployed, you can enable the follow links feature in the `followLinks` attribute of the `transferRules` element. For example:

```
<transferRules followLinks="yes" />
```

## Parameter Substitution

*Parameter substitution* is a feature that allows you to format attributes as parameters whose values you can specify on a deployment-specific basis using `iwodstart` command-line tool. You can configure any attribute value in a deployment configuration file for parameter substitution. Each time you start a deployment using `iwodstart`, you indicate the value of each attribute configured for parameter substitution. Different instances of the same deployment can have different attribute values. The parameter substitution feature turns a deployment configuration file into a template that you can customize each time you start the deployment.

To use parameter substitution, you must determine which attributes to specify as parameters, and modify them in the appropriate deployment configuration. Parameters use the following syntax:

`$parameter^`

where *parameter* is some value you will specify in conjunction with the `iwodstart` command-line tool. For example, if you wanted to designate the `area` attribute of the `sourceFilesystem` element of a particular deployment configuration as parameter `srcarea`, you would give that attribute the following value:

`area="$srcarea^"`

Every parameter entered into the deployment configuration file must include the dollar sign (“\$”) at the beginning and the carat (“^”) at the end. Otherwise, you are free to name a parameter anything you want. However, each parameter name must be unique within that deployment configuration.

After your parameters are set, you can run the deployment from the command line using the `iwodstart` command-line tool. The syntax for using `iwodstart` with parameters is:

`iwodstart deployment -k parameter=value`



For example, if you wanted to apply the value `C:\temp` to the parameter `srcarea` in the deployment configuration file *reports*, you would enter the following command at the prompt:

```
iwodstart reports -k srcarea=C:\temp
```

Note that in the command you entered, you did not include the dollar sign or the carat in the parameter name. You also did not include the value in quotation marks. However, if either the parameter or its assigned value contained a space, then the entire combined parameter and value must be placed inside of quotation marks. For example, if the value of `srcarea` is:

```
C:\Program Files\monthly
```

then you would enter:

```
iwodstart reports -k "srcarea=C:\Program Files\monthly"
```

Multiple `-k key=value` pairs may exist on the command line, for example:

```
iwodstart reports -k src=/mysource -k trg=/mytarget
```

The `iwodstart` command can only be issued on the host where the OpenDeploy server is installed. This command can be issued by anyone regardless of whether they hold an Administrator or User role. There are no authentication or authorization checks on individuals issuing command-line tools.

See “Command Line” on page 143 for more information on that `iwodstart` command.

## Deploy and Run

The *Deploy and Run* feature allows you to configure OpenDeploy to execute an external script at a specified stage of the deployment. This stage can be the deployment of a particular type or class of file or directory, or even the success of the deployment itself. Using Deploy and Run, you can configure OpenDeploy to do the following tasks automatically:

- Execute a notification script upon a failed deployment
- Run a language-checking script during deployment
- Alert you each time an executable file (with a file extension of `.exe`) is deployed

OpenDeploy supports any scripting language. The script must reside on the server where it is to be invoked. OpenDeploy will not transfer that script.

### Secure Invocation of External Applications on UNIX

The Deploy and Run scripting facility has been enhanced to launch external applications on UNIX servers using the deployment user's ID as the process owner. This applies to both sender- and receiver-side Deploy and Run invocations. This feature helps prevent a sender from launching potentially harmful operations that the user is not permitted to perform on sending and receiving systems.

Although this feature applies only to deployments running on UNIX hosts, deployments can be initiated from OpenDeploy running on either UNIX or Windows hosts. By default, the Deploy and Run script will run as the user who invoked the deployment if the `user` attribute is unspecified in the Deploy and Run `script` element.

## Requirements

Deploy and Run requires the following components:

- The presence of a customized script to be run.
- A directive indicating in what situation the script should be invoked:
  - Deployment of a specific file or filenames matching a certain pattern
  - Deployment [creation] of a specific directory
  - Before or after the actual deployment.
  - On the source or target side of the deployment
  - On success or failure of the deployment, or always

Not all of these options are applicable in combination with one another.

## Configuration

You configure Deploy and Run in the deployment configuration file. The feature is configured within the `deployNRun` element. For example:

```
<deployNRun>
  <dnrFile location="target" when="before" state="success" mask=".txt$">
    <script cmd="email_to_admin.bat" as="webmaster" where="C:\temp"
      async="no" />
  </dnrFile>
  <dnrDir location="target" when="after" state="failure" mask="exes">
    <script cmd="mail jdoe@interwoven.com < message.txt" as="webmaster"
      where="C:\temp" async="no" />
  </dnrDir>
  <dnrDeployment location="source" when="after" state="success">
    <script cmd="C:\default\main\dev\scripts" as="webmaster" where="C:\temp"
      async="yes" />
  </dnrDeployment>
</deployNRun>
```

Here is a list of child elements associated with the `deployNRun` element:

- `dnrFile` — specifies under what conditions deployed files can trigger a Deploy and Run script.
- `dnrDir` — specifies under what conditions deployed directories can trigger a Deploy and Run script.
- `dnrDeployment` — specifies under what conditions a deployment itself can trigger a Deploy and Run script.

## File-Based

You can configure OpenDeploy to begin a Deploy and Run script when a certain type or class of files are deployed. This type of Deploy and Run is known as being *file-based*. It is supported only for non-transactional deployments. You enable this feature by defining the `dnrFile` element in the Deploy and Run configuration. The `dnrFile` has the following associated attributes:

- `location` — (target) indicate where the Deploy and Run script is taking place. The only currently supported option is `target`.
- `when` — (before|after) indicate whether the script should be executed before or after the deployment of the particular file. There is no default value. You must specify one of the options.
- `state` — (success|failure|always) indicate whether the Deploy and Run script should run as a result of the success or failure of the deployment, or whether it should always run in either case. Default value is `always`.

If the `when` attribute is set to `before`, then the `state` attribute is implicitly set to `always`, because there has been no deployment yet to determine success or failure.

- `mask` — enter the regular expression to be matched against filenames to determine if the script will be executed. If you include file path separators in your mask value, you must use the path syntax supported by your OpenDeploy server.

In the following example:

```
mask=".*\\.html$"
```

any file with the file extension `.html` in the specified path will trigger the script defined within the Deploy and Run configuration.

In the following example:

```
<dnrFile
  location="target"
  when="before"
  state="always"
  mask=".*\\.exe$"
>
...
</dnrFile>
```

the Deploy and Run script, when triggered, will be located on the target host. It will occur before a file matching the value of the `mask` attribute is deployed. OpenDeploy will trigger the script whether the deployment is successful or not. (If the `when` attribute is set to `before`, the `state` attribute is implicitly set to `always` because there has been no deployment yet to determine success or failure.) The `mask` attribute value `.*\\.exe$` indicates that the script will start each time a file with the extension `.exe` is deployed.

If you are not familiar with regular expressions, consult a reference manual such as *Mastering Regular Expressions* by Jeffrey Friedl.

### Limitations

File-based Deploy and Run is not available for use with transactional deployments.

## Directory-Based

You can configure OpenDeploy to begin a Deploy and Run script when a certain type or class of directories are deployed. This type of Deploy and Run is known as being *directory-based*. It is supported only for non-transactional deployments. You enable this feature by defining the `dnrDir` element in the Deploy and Run configuration. The `dnrDir` has the following associated attributes:

- `location` — (target) indicate where the Deploy and Run script is taking place. The only currently supported option is `target`.
- `when` — (before|after) indicate whether the script should be executed before or after the deployment of the particular directory. There is no default value. You must specify one of the options.

- `state` — (success | failure | always) indicate whether the Deploy and Run script should run as a result of the success or failure of the deployment, or whether it should always run in either case. Default value is `always`.

If the `when` attribute is set to `before`, then the `state` attribute is implicitly set to `always`, because there has been no deployment yet to determine success or failure.

- `mask` — enter the regular expression specifying the directories that, when deployed, will trigger the script. You must use the path syntax supported by your OpenDeploy server. The script is triggered *only* when the directory itself is deployed on the target host. Deploying a file that resides within the directory will *not* trigger the script.

In the following example:

```
mask="cgi-bin$"
```

any directory named `cgi-bin` that is deployed will trigger the script.

In the following example:

```
<dnrDir
  location="target"
  when="after"
  state="success"
  mask="Temp$"
>
...
</dnrDir>
```

the Deploy and Run script, when triggered, will be located on the target host. It will occur after a directory matching the value of the `mask` attribute is deployed, and only if the deployment is successful. The `mask` attribute value `Temp$` indicates that the script will start each time a directory with the name `Temp` is deployed.

If you are not familiar with regular expressions, consult a reference manual such as *Mastering Regular Expressions* by Jeffrey Friedl.

## Limitations

Directory-based Deploy and Run is not available for use with transactional deployments.

## Deployment-Based

You can configure OpenDeploy to begin a Deploy and Run script when the deployment configuration itself is run. This type of Deploy and Run is known as being *deployment-based*. You enable this feature by defining the `dnrDeployment` element in your Deploy and Run configuration. The `dnrDeployment` element has the following associated attributes:

- `location` — (`source` | `target`) indicate whether the Deploy and Run script is taking place on the source or target host. There is no default value. You must specify one of the options.
- `when` — (`before` | `after`) indicate whether the script should be executed before or after the deployment occurs. There is no default value. You must specify one of the options.

Evaluation of the `area` and `previousArea` attribute values in TeamSite comparison deployments with respect to the latest and next-to-latest editions, occurs *before* the running of any Deploy and Run scripts.

- `state` — (`success` | `failure` | `always`) indicate whether the Deploy and Run script should run as a result of the success or failure of the deployment, or whether it should always run in either case. Default value is `always`.

If the `when` attribute is set to `before`, then the `state` attribute is implicitly set to `always`, because there has been no deployment yet to determine success or failure.

In the following example:

```
<dnrDeployment
  location="source"
  when="after"
  state="failure"
>
...
</dnrDeployment>
```

the Deploy and Run script triggers when this particular deployment configuration is run. The script originates at the source host, and is executed after the occurrence of a failed deployment.

## Deploy and Run Scripting

The Deploy and Run script is defined by the `script` element. Once the script is in place, the `script` element will integrate it into the Deploy and Run configuration. The `script` element has the following attributes:

- `cmd` — enter the full path to the command which OpenDeploy should run, if triggered, as well as any accompanying flags or options. If you specify the `where` attribute (see below) you may be able to use the value “./” on UNIX, or no path specification on Windows. You can also specify an executable invocation line. For example:

```
cmd="C:\bin\email_to_admin.bat -user jdoe@interwoven.com" or
```

```
cmd="/bin/mail jdoe@interwoven.com < /tmp/message.txt"
```

If the command you are going to run requires a scripting engine, the scripting engine must be on the `PATH` of the user (or system, on Windows) who will be running the script or specified with a full path). For example:

```
cmd="/bin/sh /usr/local/bin/email_to_admin.sh -u jdoe@interwoven.com"
```

*or*

```
cmd="/usr/local/bin/iwperl /path/to/script.pl"
```

- `where` — enter the location to where OpenDeploy must go before executing the script. For example:

```
where="/tmp" or
```

```
where="C:\temp"
```

You must use the path syntax supported by your OpenDeploy server. Forward slashes (“/”) can be used for either Windows or UNIX hosts, while backslashes (“\”) are only permitted on Windows hosts.

The `where` attribute is optional. If you do not specify a value, the process takes place in the root directory.

- `as` (UNIX only) — enter a different user name or user ID under which you will run the script. This option allows you to run the script as a different user. In the following example:

```
as="rroe" or
```

```
as="110"
```

you can run the script as `rroe` or its user ID `110` rather than your regular user name. By default, the script runs as the user who invokes OpenDeploy. The user who will need to be root for most purposes.

- `async` — indicate whether or not to run the script asynchronously. Exercise caution when using this mode, as it could cause many scripts to be run simultaneously. The output from scripts run asynchronously is not captured. Default value is `no`.

In the following example:

```
<dnrDeployment
  location="source"
  when="after"
  state="always"
>

<script
  cmd="/bin/sh /usr/local/bin/email_to_admin.sh"
  where="/tmp"
  async="yes"
/>

</dnrDeployment>
```

the Deploy and Run configuration will trigger the `/bin/sh /usr/local/bin/email_to_admin.sh` script (which sends deployment confirmation email messages to the OpenDeploy administrator) after the deployment has completed, whether successful or not. The script is also allowed to run asynchronously.

## Deploy and Run Script Logging

The standard output generated by Deploy and Run scripts is sent to a special XML log in OpenDeploy. This log is an in-memory buffer that keeps track of all events from OpenDeploy and is parsed to provide the output found in the log file. The log file itself is not affected by the output of Deploy and Run scripts.

The following steps take place whenever Deploy and Run calls a script:

1. `stdin` is set to receive an XML representation of the OpenDeploy in-memory log file in its current state.
2. The script executes and the results are sent to `stdout`.
3. The receiver XML log is transferred to the sender.

In this manner, future scripts can parse the output of past scripts. For example, a script might extract information about which files were deleted during the last deployment.

The XML representation of the OpenDeploy in-memory log file has the following DTD:

```
<!DOCTYPE log [
  <!ELEMENT log ANY>
  <!ATTLIST log target CDATA "">
  <!ATTLIST log action CDATA "0">
  <!ATTLIST log date CDATA "0">
  <!ATTLIST log result CDATA "0">
  <!ATTLIST log response CDATA "">
  <!ELEMENT log_element ANY>
  <!ATTLIST log_element target CDATA "">
  <!ATTLIST log_element action CDATA "0">
  <!ATTLIST log_element date CDATA "0">
  <!ATTLIST log_element result CDATA "0">
  <!ATTLIST log_element response CDATA "">
]>
```

Within Deploy and Run log entries, the following conditions apply:

- The term `target` specifies the name of the file or directory deployed.
- The term `action` is one of the following:

#	Name	Description
0	UNDEFINED	Undefined action
1	LOG_ELEMENT_DEPLOY_FILE	Deploy a file
2	LOG_ELEMENT_DEPLOY_DIRECTORY	Deploy a directory
3	LOG_ELEMENT_SUMMARY	Contains summary of events
4	LOG_ELEMENT_RUN_SCRIPT	Run a script
5	LOG_ELEMENT_DELETE_FILE	Delete a file
6	LOG_ELEMENT_DELETE_DIRECTORY	Delete a directory

The term “date” is represented as the number of [non-leap] seconds since 00:00:00 UTC January 1, 1970.

The term `result` is one of the following:

#	Name	Description
-2	LOG_ELEMENT_ABORT	The result of the action was to signal the deployment to do a hard abort in which the deployment stops immediately. If the deployment was transactional, the files on the production server will be returned to their original state.
-1	LOG_ELEMENT_ERROR	The action returned an error.
0	LOG_ELEMENT_OK	The action returned a satisfactory result and the item should be printed to STDOUT in all cases.
1	LOG_ELEMENT_OK1	The action returned a satisfactory result which should be printed to STDOUT if the verbosity level is set to <code>normal</code> .
5	LOG_ELEMENT_OK5	The action returned a satisfactory result which should be printed to STDOUT if the verbosity level is set to <code>verbose</code> .

The term `response` is the text response for the action.

For example, a log file might contain the following line:

```
<log_element target="/tmp/Branch1/src/pmt" action="2"
date="925263642" result="0" response="" />
```

indicating that the target was /tmp/Branch1/src/pmt, the action was to deploy a directory, the result was satisfactory, and there was no text response for this action.

### Capturing Deploy and Run Script Log Entries

To capture and record Deploy and Run script log entries, you can use either of the following methods:

- Configure the Deploy and Run script to produce its own log file.
- Configure another Deploy and Run script to re-pipe STDIN to another file.

In Figure 50, one Deploy and Run script is configured to run before a deployment, and a second Deploy and Run script is configured to run after the deployment. The second script is responsible for transferring information from STDIN to a separate log file (`foo.log`).

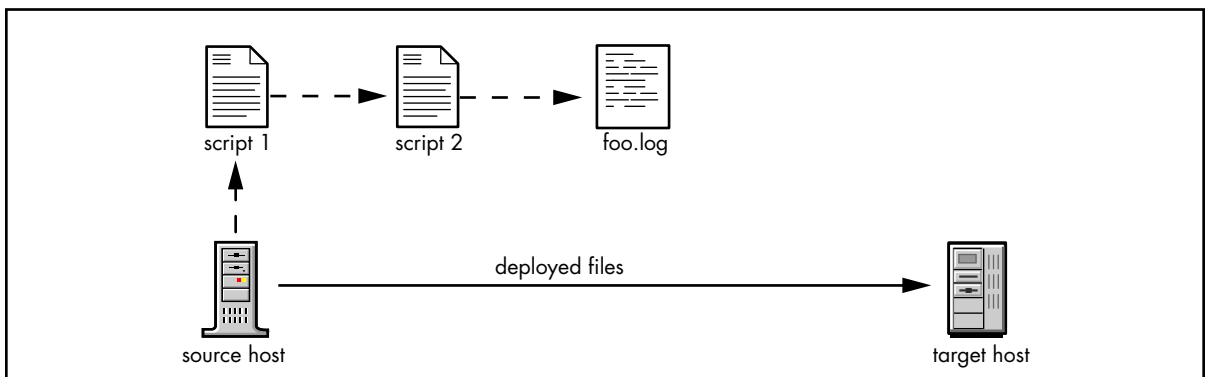


Figure 50: Deploy and Run Logging

The first Deploy and Run script (`script 1`) has the option to read from STDIN and extract details regarding the deployment. This script can then send output to STDOUT which will be captured by OpenDeploy's in-memory log. The second Deploy and Run script (`script 2`) parses STDIN which is in XML format. This script can retrieve details regarding the deployment, and the output from the first Deploy and Run script. These details can then be written to a log file (`foo.log`).

## Communicating Status to OpenDeploy

A DNR script can send a return code to OpenDeploy by printing the following:

```
<response code="-2"/>\n
```

to STDOUT. A value of -2 indicates that the result of the action was to signal the deployment to do a hard abort in which the deployment stops immediately. If the deployment was transactional, the files on the production server will be returned to their original state.

## Deploying to a Package File Using Deploy and Run

If it is impossible or impractical to deploy files directly to your target hosts, perhaps because of a firewall or some other obstruction, OpenDeploy can deploy files into a package file on another host, or even the source host itself. You can transport the file to the target hosts using an approved means, and install the deployed files directly on the targets. OpenDeploy uses the Deploy and Run feature as the basis for creating package files.

In Figure 51, a direct transmission of files between the source host and the target host is not possible. The OpenDeploy administrator configures a deployment to write the deployed files to a package file, such as a .tar or .zip file, on the source host. That package file is then copied to a transportable medium, such as a tape, and is subsequently manually installed on the target host.

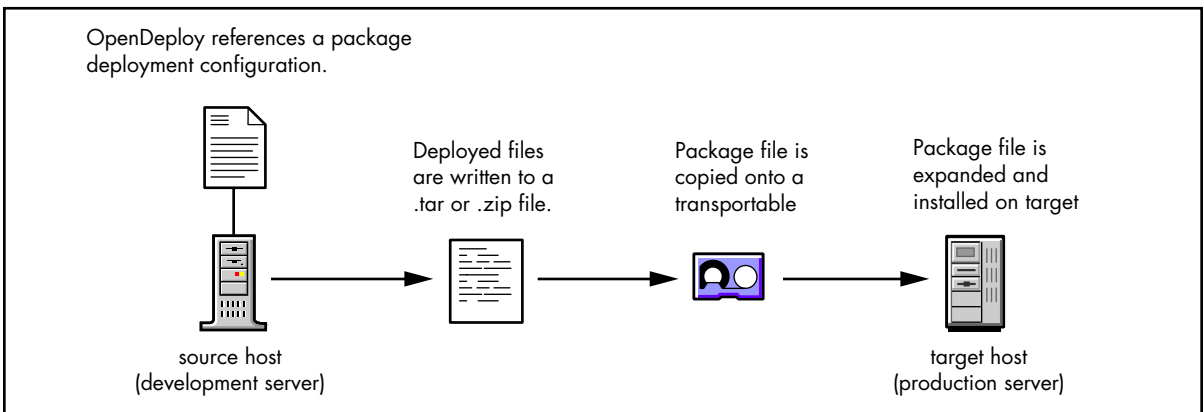


Figure 51: Package Deployment

To create a package file, follow these steps:

1. Configure a deployment that deploys files to the source host itself. This involves the following tasks:
  - Specifying your source host in the nodes configuration file.
  - Adding your host and target directory to the `allowedHosts` and `allowedDirectories` elements in your base server configuration file.
2. Configure the deployment with a Deploy and Run script that writes the deployed files to a package file after a successful deployment. On an OpenDeploy host running on a UNIX host, this would appear in your deployment configuration as the following:

```
<dnrDeployment location="target" when="after" state="success">  
  <script cmd="tar cvf package_file.tar target_area" async="no" />  
</dnrDeployment>
```

where *package\_file.tar* is the name of the package file and *target\_area* is the path to the location where the deployed files will reside after completing the deployment.

On an OpenDeploy host running on Windows, you would substitute a Windows packaging command for the `cmd` attribute.

## Encryption

OpenDeploy provides two methods of encryption:

- Weak (40-bit) symmetric key file-based encryption
- Strong (up to 168-bit) asymmetric key encryption using Secure Sockets Layer-based (SSL) encryption

These types of encryption are mutually exclusive and cannot be used in conjunction with one another. Be sure not to include attributes for both types of encryption in the same configuration.

Encryption can be specified both at the OpenDeploy base server and receiver level, and at the individual deployment configuration level. Encryption settings specified in the deployment configuration level will automatically override any encryptions settings in the server configuration.

### Symmetric Key Encryption

OpenDeploy provides 40-bit encryption support for content transfers through referencing an encryption algorithm key file specified in the base server or receiver configuration file. OpenDeploy symmetric key deployment provides basic encryption support with minimal performance impact on content deployment. However, symmetric 40-bit encryption is breakable by brute force attack with a modest amount of computing power and is potentially vulnerable to unauthorized users with the same symmetric key who can intercept data in transit.

#### Configuring OpenDeploy for Symmetric Encryption

The path and filename can be specified in the `keyFile` attribute of the `localNode` element in either the base server, receiver, or deployment configuration files. If you configure your base server or receiver configuration files for symmetric configuration, but do not specify encryption in the specific deployment configuration, then by default the deployment will use symmetric encryption. Whatever you specify for symmetric encryption in the deployment configuration will override any settings in the base server or receiver configuration files.

The `keyFile` attribute specifies the absolute path to the symmetric key. Here is an example of the OpenDeploy server *mars* being configured for symmetric key encryption:

```
<localNode
  host="mars"
  keyFile="/local/OpenDeploy/conf/keyfile.txt"
/>
```

### Using Symmetric Encryption with Reverse Deployments

If you are performing a reverse deployment using symmetric key encryption, you must include the path to the symmetric key file residing on the reverse-source host (normally the receiving host) in the deployment configuration. This is the same location as specified in the base server or receiver configuration file of the reverse-source host. This differs from a forward deployment, where the configuration would include the path to the key file where it resides on the source host. The deployment configuration must include the path syntax of the reverse-source host. The path to the symmetric key file is defined in the `keyFile` attribute of the `localNode` element.

In the following example, the source host *mars* has the base server software installed and is running on UNIX. The target host *venus* has the receiver software installed and is running on Windows. In a typical forward deployment using symmetric key encryption, the `localNode` element in the deployment configuration residing on *mars* would be:

```
<localNode
  host="mars"
  keyFile="/local/OpenDeploy/conf/keyfile.txt"
/>
```

and the `localNode` element in *venus*' receiver configuration file would be:

```
<localNode
  host="venus"
  keyFile="C:\encryption\keyfile.txt"
/>
```

In a reverse deployment involving these two hosts, the `localNode` element in the reverse deployment configuration would be:

```
<localNode
  host="mars"
  keyFile="C:\encryption\keyfile.txt"
/>
```

and the `localNode` element in the base server configuration file on *mars* would be:

```
<localNode
  host="mars"
  keyFile="/local/OpenDeploy/conf/keyfile.txt"
/>
```

## Asymmetric Key Encryption

OpenDeploy provides up to 168-bit asymmetric key encryption support for secure content transfers. This deployment option uses a certificate authority (provided with OpenDeploy) to ensure that all content transfers occur only between known and trusted sources. The OpenDeploy asymmetric key algorithm uses public key exchange to authenticate destination servers, and can use any one of a number of algorithms to encrypt content prior to deployment.

OpenDeploy asymmetric key deployment offers the strongest possible encryption support and is unlikely to be broken by any brute force attack. Furthermore, when used in conjunction with digital certificates for initial server authentication, asymmetric key encryption is not vulnerable to a “man-in-the-middle” attack. Asymmetric key encryption provides the strongest possible security support in exchange for a small performance cost.

The use of 168-bit encryption is available only within the United States of America. For more information about encryption and ciphers, consult a cryptography reference manual such as *Applied Cryptography* (Bruce Schneier, ISBN 0-471-11709-9). You can also set up asymmetric encryption to provide less than full 168-bit security.

## Prerequisite Tasks

OpenDeploy implements asymmetric encryption at both the source and target host level, collectively known here as *peers*. Each peer host running OpenDeploy software, either as a source host or a target host, contains encryption configuration information within the base server or receiver configuration file's `localNode` element. For asymmetric encryption, OpenDeploy uses the OpenSSL implementation of the Secure Sockets Layer (SSL). Before you use SSL, you must generate certificates and keys on both the source and target hosts. This involves the following tasks:

- Setting up the certificate authority
- Generating the certificate

These tasks create two unique public and private key pairs that are signed by the same certificate authority in order to enable asymmetric encryption. One key pair will be copied to the source host. The other key pair will be copied to the target host. These tasks are required regardless of what level of asymmetric encryption you want to use. Either the source or target host has the ability to request a verification of the certificate authority before the deployment can occur. See “Configuring OpenDeploy for Asymmetric Encryption” on page 287 for more information.

The *certificate authority* consists of a set of programs used to generate public and private key pairs and a database that contains state information. The programs are installed in the following location:

`od-home/bin`

By default, the database is contained in the directory where the programs are run. If future public and private key pairs are created using a different certificate authority, OpenDeploy will not be able to deploy to or from a host with keys created by an older certificate authority.

## Setting Up the Certification Authority

To set up the OpenSSL RSA certificate authority, follow these steps:

1. Verify that the `od-home/bin` directory is included in the `PATH` environment variable.
2. Navigate to the `od-home/bin` directory.
3. Create a seed file for the random number generator.

OpenSSL uses a pseudo random number generator (PRNG) in the process of generating the keys. The PRNG wants to be seeded with a satisfactory amount of genuine random data, so it does not generate predictable keys. You can copy any file sufficiently large into a `.rnd` file located at your HOME directory to make it a seed file. On Windows, the HOME directory is a User variable value, and can vary.

For UNIX, you can also enter the following command at the prompt:

```
ypcat passwd > ~/.rnd
```

If you do not want the `.rnd` file being in the root (`/`) directory (which is what happens when running `CA.sh` from root), then you can change the `RANDFILE` parameter (`$ENV::HOME/.rnd`) in the `openssl.cnf` file to reflect your preferred location.

4. Install the new certificate authority by entering the following command (depending on the platform) at the prompt:
  - Windows — **CA.bat -newca** *or*
  - UNIX — **CA.sh -newca**

The `CA.bat -newca` command takes another parameter for using a different certificate authority. For example:

```
CA.bat -newca demoCA\certs\cacert.pem
```

while the `CA.sh -newca` command reads the parameter during the execution of the script.

5. Enter the appropriate information when prompted for the following:

- Country Name
- State or Province Name
- Locality Name
- Organization Name
- Organization Unit Name
- Command Name
- Email Address

The information you enter will be incorporated into the certificate authority. These prompts are defined by the `openssl.cnf` configuration file, and can be changed according to your requirements.

### Generating a Certificate

You must generate an RSA Certificate twice, once for the source host and once for the target host.

To generate an RSA Certificate for an OpenDeploy host, follow these steps:

1. Navigate to the `od-home/bin` directory.
2. Generate a new certificate and key by entering the following command (depending on the platform) at the prompt:
  - Windows — **CA.bat -certall** *or*
  - UNIX — **CA.sh -certall**
3. Enter the appropriate information when prompted for the following:
  - Country Name
  - State or Province Name
  - Locality Name
  - Organization Name
  - Organization Unit Name
  - Command Name
  - Email Address



- A challenge password
- (Optional) Company name

You cannot have two or more certificates with the exact same information. Each certificate must be unique.

The information you enter will be incorporated into the certificate authority. These prompts are defined by the `openssl.cnf` configuration file, and can be changed according to your requirements.

4. Answer yes at the prompts to sign and then commit the certificate:

```
Sign the certificate? [y/n]: y
```

```
1 out of 1 certificate requests certified requests certified, commit?  
[y/n]: y
```

5. Copy the generated keys to the appropriate locations, depending on which peer host the certificate/key pair is intended.

A good place to store certificates and keys is:

```
od-home/cert
```

You must create this directory manually, it is not generated during the installation. You might also want to rename the keys to reflect their roles in the deployment cycle, for example, name your source key files `odsrckey.pem` and `odsrccert.pem`, and your target key files `odtgtkey.pem` and `odtgtcert.pem`.

At the conclusion of these steps a private key file called `newreq.pem` and a certificate file called `newcert.pem` are generated.

## Configuring OpenDeploy for Asymmetric Encryption

After you generate and sign the certificates as described in the preceding sections, you must configure OpenDeploy to use asymmetric encryption. Encryption values are specified in the `localNode` element of the base server or receiver configuration files of the OpenDeploy host. If you specify asymmetric encryption in these configuration files, then all deployments will be encrypted this way. You must have the following attributes and their values specified within the `localNode` element:

- `sslCertificate` — enter the absolute path to the SSL public key certificate.
- `sslPrivateKey` — enter the absolute path to the SSL private key certificate.
- `sslCaCertificate` — enter the absolute path to the certificate authority. This allows OpenDeploy to authenticate the source from which the public and private key pairs for the source and target hosts are derived.
- `sslVerifyPeer` — (`none` | `request` | `required`) indicate which of the following conditions apply in regards to the verification that the certificate authority for each public and private key pairs comes from the same source. This source is the value specified in the `sslCaCertificate` attribute.
  - `none` — no verification is performed. This is the default value.
  - `request` — verification is performed if the certificate/key pair exists on the peer of the host making the authentication request before the deployment can occur.
  - `require` — verification must be performed, and the certificate/key pair must exist on the peer of the host making the request before the deployment can occur.

Here is an example of how the `localNode` element and its encryption-related attributes might be configured for asymmetric encryption:

```
<localNode
  host="mars"
  ssl Certificate="od-home/usr/opendeploy/conf/odcert.pem"
  sslPrivateKey="od-home/usr/opendeploy/conf/odkey.pem"
  sslCaCertificate="od-home/bin/demoCA/certs/cacert.pem"
  sslVerifyPeer="request"
/>
```



## Asymmetric Encryption During Down-Rev Deployments

If you are performing a deployment from an OpenDeploy 5.5.1 source to an OpenDeploy 4.5.2 target using asymmetric encryption, you must specify the path to the OpenDeploy 4.5.2 certificate and key files in the deployment configuration, rather than the path to the OpenDeploy 5.5.1 certificate and key files. For example:

```
<localNode
  host="mars"
  ssl Certificate="od-4.5.2-home/usr/opendeploy/conf/odcert.pem"
  sslPrivateKey="od-4.5.2-home/usr/opendeploy/conf/odkey.pem"
  sslCaCertificate="od-4.5.2-home/bin/demoCA/certs/cacert.pem"
  sslVerifyPeer="request"
/>
```

When OpenDeploy recognizes that the target is an OpenDeploy 4.5.2 host, it will convert the file's XML-based format file into a format compatible with OpenDeploy 4.5.2 and pass the control to the OpenDeploy 4.5.2 client. This configuration file will be saved in the following location on the source host:

```
od-home/lib
```

Installation of OpenDeploy 4.5.2 on the OpenDeploy 5.5.1 host is required to perform down-rev deployments. See “Deploying to OpenDeploy Downward Revision Targets” on page 89 for more information.

## Ciphers

You can specify various ciphers to use in encryption. During a connection, the OpenDeploy source and target hosts will negotiate over which cipher to use. During the negotiation phase, OpenDeploy selects the highest priority cipher that both source and target hosts support. The use of ciphers is specified by the presence of the `sslCiphers` attribute in the `localNode` element, located in the base server or receiver configuration file. For example:

```
sslCiphers="cipherlist"
```

where *cipherlist* contains one or more ciphers, ranked left to right from highest priority to lowest priority, separated by colons (":"). For example:

```
sslCiphers="EDH-DSS-DES-CBC3-SHA:EXP-EDH-DSS-DES-CBC-SHA"
```

OpenDeploy allows you to use the following types of ciphers:

- No-authentication ciphers:
  - None
- Low-strength ciphers:
  - RC4-64-MD5
  - EXP1024-DHE-DSS-RC4-SHA
  - EXP1024-RC4-SHA
  - EXP1024-DHE-DSS-DES-CBC-SHA
  - EXP1024-DES-CBC-SHA
  - EXP1024-RC2-CBC-MD5
  - EXP1024-RC4-MD5
  - EDH-RSA-DES-CBC-SHA
  - EDH-DSS-DES-CBC-SHA
  - DES-CBC-SHA
  - DES-CBC-MD5
  - EXP-EDH-RSA-DES-CBC-SHA
  - EXP-EDH-DSS-DES-CBC-SHA
  - EXP-DES-CBC-SHA
  - EXP-RC2-CBC-MD5
  - EXP-RC4-CBC-MD5
  - EXP-RC2-CBC-MD5
  - EXP-RC4-MD5
- Medium-strength ciphers:
  - DHE-DSS-RC4-SHA
  - RC4-SHA
  - RC4-MD5
  - RC2-CBC-MD5
  - RC4-MD5



- High-strength ciphers:
  - EDH-RSA-DES-CBC3-SHA
  - EDH-DSS-DES-CBC3-SHA
  - DES-CBC3-SHA
  - DES-CBC3-MD5

If `sslCiphers` is not specified, the default cipher value is:

`EXP-RC4-MD5`

The following example adds a 168-bit cipher and a low-strength cipher to the asymmetric key encryption code created in “Configuring OpenDeploy for Asymmetric Encryption” on page 287:

```
<localNode
  host="mars"
  sslCertificate="od-home/usr/opendeploy/conf/odcert.pem"
  sslPrivateKey="od-home/usr/opendeploy/conf/odkey.pem"
  sslCaCertificate="od-home/bin/demoCA/certs/cacert.pem"
  sslVerifyPeer="request"
  sslCiphers="EDH-DSS-DES-CBC3-SHA:EDH-DSS-DES-CBC-SHA"
/>
```

## Redeploying Legacy Web Sites

If you are using OpenDeploy in conjunction with TeamSite, you can make a copy of a set of deployed files and store it for later use. This feature is handy if you need to “roll back” an existing Web site to a previous version, or recreate a Web site as it once looked. When a set of Web files on a TeamSite server is complete and ready to deploy, make a TeamSite edition of those files. Include the date or some other identifying element in the edition name. Later, if you need to recreate a legacy Web site, you can deploy that saved edition.

You can deploy the TeamSite edition using the TeamSite comparison method. Specify the `area` attribute value as the path to the TeamSite edition you want to deploy. Specify a TeamSite area that contains no value for the `previousArea` attribute, such as the initial edition that is generated for the TeamSite base branch. When the deployment is run, all the files in the TeamSite edition you need to restore your legacy Web site will be redeployed to the target host. See “TeamSite Comparison Deployments” on page 214 for more information.



## Chapter 6

# Composing Deployments

---

This chapter describes the Deployment Configuration composer, and how to use it to create complete functional deployment configurations through the OpenDeploy user interface. Although the ability to write and understand XML code is not required to create and edit deployment configurations using the Deployment Configuration Composer, it is recommended that you review the features and functionality described in Chapter 4, “Deployment Configurations” prior to creating and editing any new or existing deployment configurations.

## Deployment Configuration Composer

The Deployment Configuration Composer is a tool within the OpenDeploy user interface for creating new deployment configurations and editing existing ones. You can author or edit the configuration of any deployment that resides in the *od-admin/conf* directory, and that conforms to XML-based structure used in OpenDeploy 5.x deployment configurations.

The text boxes, lists, and option buttons that appear in the Deployment Configuration Composer interface correspond to elements and attributes in the deployment configuration file. The values you input are added to the deployment configuration file. Any existing deployment configuration can be edited and updated using the Deployment Configuration Composer.

To access the Deployment Configuration Composer, follow these steps:

1. Select **Configurations > View Configurations** to display the Deployment Configuration window.
2. Select the OpenDeploy host on which the deployment configuration will reside from the **Selected Host** list.
3. Click **New** to open a new browser window containing the containing the Deployment Configuration Composer (Figure 52).

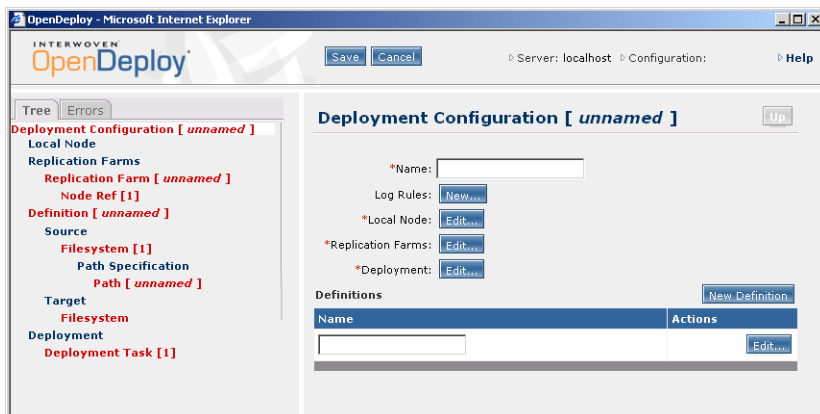


Figure 52: Deployment Configuration Composer

To edit an existing deployment configuration, select that configuration from the **Deployment** list and click **Edit** to open the Deployment Configuration Composer displaying the deployment configuration you selected.

## Tree and Errors Tabs

The Deployment Configuration Composer contains Tree and Errors tabs on the left that display either a tree of configuration elements from which you can select, or a list of errors that display omissions of required information in your deployment configuration (Figure 53).

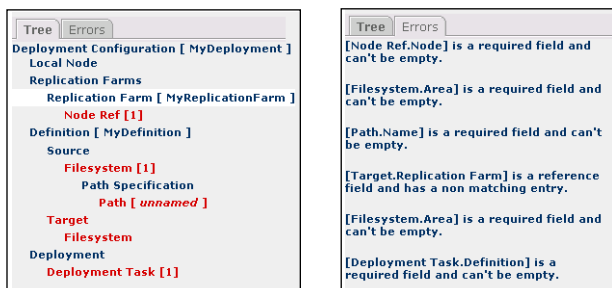


Figure 53: Tree and Error Tabs

## Tree Tab

The Tree tab displays the required and optional elements that make up the deployment configuration. Those elements in red are required to be completed before the configuration is done. After the information corresponding to a red element entry has been provided, the entry will turn blue.

Values contained in brackets ([ ]) reflect either a unique name value you assign them, for example:

```
Deployment Configuration [MYCONFIGURATION]
```

or the numbered occurrence of that elements, for example:

```
Filesystem [2]
```

When you complete adding information to a window in the Details pane, you can display the previous window by clicking the **Up** button. You can also display another window by selecting its entry in the Tree pane. When you provide names for elements, such as the definition element, that name is displayed next to its element in the Tree tab.

## Errors Tab

The Errors tab displays error messages associated with any elements, attributes, or values that are missing or incorrect. You then must complete the required information and save the file again. When you save a deployment configuration, the Errors tab will automatically display any associated errors.

## Details Pane

Selecting an entry in the Tree tab displays the corresponding window in the Details pane on the right of the browser window. Each window contains text boxes, buttons, drop-down lists, and other features which correspond to attributes in the deployment configuration. In some cases you must enter values for these attributes. In other cases, you have the option of providing a value, or allowing OpenDeploy to use its default values. Those attributes with a red asterisk (\*) to the left denote required attributes for you to complete. In some cases, you can access a window in the Details pane either by selecting an entry in the tree, or by clicking a button in another Details pane window.

## Types of Deployment Configuration Settings

Deployment configuration settings fall under the following categories:

- Global settings — configuration applies to the deployment as a whole. These settings include logging, nodes and node farms, transactional capability, and Deploy and Run scripting.
- Definition settings — configuration applies to the specified definition element, which consists of a source/target pairing and its associated rules. These settings include the source and target areas, filters, and deployment rules.

### Global Deployment Settings

Global deployment settings apply to all sources, targets, and deployed files included in the deployment. The following required tasks apply:

- Name the deployment.
- Verify or change the source host name.
- Name your default node farm element.
- Assign one or more target hosts listed in your nodes configuration file to each node farm.
- Indicate whether or not the deployment is transactional.

You can apply perform these optional tasks:

- Set your log rules.
- Configure encryption.
- Add and name any additional node farm elements.
- Enter the name of the deployment that your target host will run after receiving your deployed files. Also indicate whether the target host will invoke a new deployment if your deployment is not successful. These tasks are required for multi-tiered deployments only.
- Assign Deploy and Run capabilities as necessary, including those scripts that trigger upon the deployment of particular files, directories, or deployments.

## Definition Settings

A deployment configuration contains one or more definition elements. Each definition element contains additional elements and attribute values that identify the source locations where deployed files originate, and the target location where those deployed files are received. For comparison-based deployments, those file locations participating in the deployment also are specified. Any rules establishing the deployment eligibility criteria of the files are also contained within the definition.

The following required tasks apply to each instance of the definition element in your deployment configuration:

- Name the default definition element.
- Indicate whether the definition is a forward or reverse deployment.
- Specify the type of location (file system or TeamSite area) where the files to be deployed reside on the source host.
- Enter the path to the source file location, including previous TeamSite area or file list path.
- Enter any subdirectories within the specified source file location that further define where the source files reside. If you do not wish to further refine the area specification, you must enter the value “.” to indicate that no subdirectory is specified.
- Enter the path to the target file location.

You can also apply these optional tasks:

- Add and name any additional definition elements.
- Add and configure any file path or pattern exclusion rules, and whether or not to follow symbolic links during the deployment.
- Configure any target area overrides for deployments with multiple source area locations. Apply any transfer, comparison, or permission rules to those files deployed to this alternate area.
- Add and configure any additional source file locations.
- Configure any comparison, transfer, or permission rules for the deployment.
- Configure any source- or target-side filters.

## Creating a New Configuration

The following section leads you through the process of creating a new deployment configuration using the Deployment Configuration Composer. Each major task is separated to make the procedure easier to learn.

To create a new deployment configuration using the Deployment Configuration Composer, follow these steps:

1. Select **Configurations > View Configurations** to display the Deployment Configuration window.
2. Select the OpenDeploy host on which your new deployment configuration will reside from the **Selected Host** list.
3. Click **New**. A new browser opens containing the Deployment Configuration Composer. The Deployment Configuration window (Figure 54) is displayed within the Deployment Configuration Composer. Here is where you will begin creating your new deployment configuration.

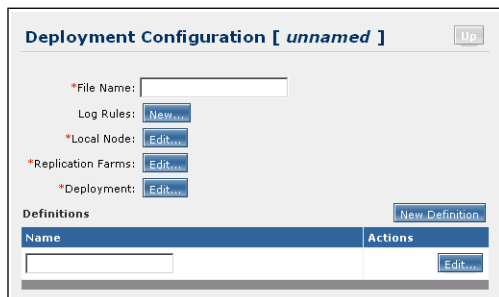


Figure 54: Deployment Configuration Window

## Naming the Deployment Configuration

Enter the file name of the deployment configuration in the **File Name** box.

You can enter a file name of any length up to the limit supported by the host operating system. Deployment configurations residing on UNIX hosts cannot have spaces in the file names. Those running on Windows hosts may contain spaces. It is not necessary to include the `.xml` extension in the file name you enter. The Configuration Composer will automatically add that extension when you save the file.

## Specifying the Log Rules

The log rules in a deployment configuration determine the maximum size a log file can grow before that file is closed to new entries, and a new log file is started. This is known as the rollover threshold. You can also specify whether you want the logging levels to be normal or verbose. See “Logging” on page 155 for a complete description on OpenDeploy logging.

The default settings for Log Rules comes from the base server and receiver configuration files, however you can provide specific values for your deployment by clicking the **New** button associated with the **Log Rules** to display the Log Rules window (Figure 55).



Figure 55: Log Rules Window

You can also select the **Log Rules** entry in the Tree. Here you can set your own maximum log file size and logging level. You can input the following values in the Log Rules window:

- **Maximum Size** box — enter the rollover threshold value. You can specify different byte measurements in the value, including megabytes (mb), kilobytes (kb), and bytes (b). For example:

10mb *or*

10000kb *or*

10000000b

If you enter a numerical value to fail to provide the measurement indicator, OpenDeploy will default to bytes (b). However, the minimum allowable size is 100 KB, or 102400 bytes. Setting a value smaller than this will be overridden with the default minimum when the deployment is run. See “Logging Configuration Settings” on page 171 for more information on OpenDeploy default logging values and rules.

- **Level** options — select the level and type of logging OpenDeploy will perform:
  - **verbose** — logs a high level of detail on deployment events as they occur. This logging level is best suited for troubleshooting deployment problems or evaluating deployment performance. Verbose logging can create very large log files. This is the default logging level.
  - **normal** — logs standard status and error messages. In most cases, this level of logging provides sufficient detail to meet your needs.

Click **Up** when you are done with the Log Rules window to return to the Deployment Configuration window.

Specifying logging rules in the Deployment Configuration Composer is optional. If no logging levels are specified, OpenDeploy will use the logging levels present in the base server configuration if present, or use the following default settings:

- **Maximum Size** (rollover threshold): 32 MB
- **Level:** Verbose

See “Logging” on page 85 for more information on setting base server logging.

## Verifying or Changing the Source Host Name

Your deployment configuration must include the name of your OpenDeploy host server. This name can either be its fully qualified DNS server name or its IP address. Click the **Edit** button associated with the **Local Host** in the Deployment Configuration window to display the Local Node window (Figure 56).

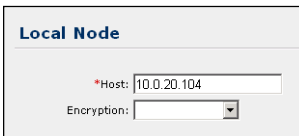


Figure 56: Local Node Window

By default, your host’s name will be displayed in the **Host** box. However, you can change that value to another OpenDeploy host if you want. An example of this would be if you are authoring a deployment configuration you intend to use on another OpenDeploy host.

## Specifying Deployment Encryption

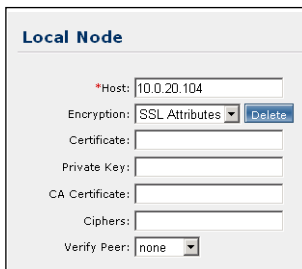
Also present in the Local Node window (Figure 56) is your deployment encryption settings. If you want to use encryption with your deployment configuration, you must specify it here in the **Encryption** list.

- **SSL Attributes** — symmetric key encryption that uses the secure sockets layer (SSL) key certificate.
- **Key File** — 40-bit symmetric encryption that uses a key file.

If you select an encryption option, the following additional configuration attributes for that option will be displayed in the window for you to complete. See “Encryption” on page 280 for more information on how encryption in OpenDeploy works.

### SSL Attributes

The following attributes are displayed when you select the **SSL Attributes** encryption option (Figure 57):



The screenshot shows the 'Local Node' configuration window. The 'Host' field contains '10.0.20.104'. The 'Encryption' dropdown menu is set to 'SSL Attributes', with a 'Delete' button next to it. Below this, there are five empty text input fields labeled 'Certificate:', 'Private Key:', 'CA Certificate:', 'Ciphers:', and 'Verify Peer:'. The 'Verify Peer:' dropdown menu is currently set to 'none'.

Figure 57: SSL Attributes

- **Certificate** box — enter the absolute path to the SSL public key certificate. This attribute is required for using asymmetric key encryption.
- **Private Key** box — enter the absolute path to the SSL private key certificate. This attribute is required for using asymmetric key encryption.
- **CA Certificate** box — enter the absolute path to the certificate authority. This allows OpenDeploy to authenticate the source from which the public and private key pairs for the source and target hosts are derived. This attribute is required for using asymmetric key encryption.

- **Ciphers** box — enter the SSL ciphers to use. Multiple ciphers must be separated by colons (":"). For example:

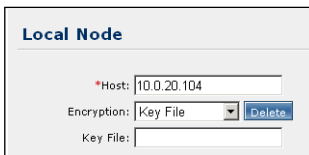
```
EDH-DSS-DES-CBC3-SHA:EXP-EDH-DSS-DES-CBC-SHA
```

This attribute is optional. Use of asymmetric encryption does require the use of ciphers.

- **Verify Peer** list — select which of the following conditions apply in regards to the verification that the certificate authority for each public and private key pairs comes from the same source. This source is the value specified in the **sslCaCertificate** attribute.
  - **none** — no verification is performed. This is the default value.
  - **request** — verification is performed if the certificate/key pair exists on the peer of the host making the authentication request before the deployment can occur.
  - **require** — verification must be performed, and the certificate/key pair must exist on the peer of the host making the request before the deployment can occur.

## Key File

The following attributes are displayed when you select the **Key File** encryption option (Figure 58):



The screenshot shows a window titled "Local Node". Inside, there is a "Host:" label followed by a text box containing "10.0.20.104". Below that is an "Encryption:" label followed by a dropdown menu showing "Key File" and a "Delete" button. At the bottom is a "Key File:" label followed by an empty text box.

Figure 58: Key File Attributes

- **Key File** box— specifies the absolute path to the key file that provides the weak 40-bit symmetric encryption. For example:

```
C:\secure\MyKeyFile.txt or
```

```
/secure/MyKeyFile.txt
```

Click **Up** when you are done with the Local Node window to return to the Deployment Configuration window.

## Naming the Replication Farm Element

The replication farm is an element in the deployment configuration that represents a collection of one or more target hosts. You must give each replication farm a unique name.

Click the **Edit** button associated with the **Replication Farms** to display the Replication Farms window (Figure 59).

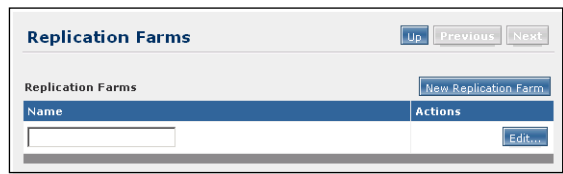


Figure 59: Replication Farms Window

Here you must enter a unique name for your replication farm set in the **Name** box, for example:

MYREPLICATIONFARM

If you want more than one replication farm in your deployment configuration, click **New Replication Farm** to add another **Replication Farms** entry. You must provide a unique name for each replication farm in your deployment configuration.

## Adding Target Host Nodes to the Replication Farm Element

After you have named your replication farm elements, you must assign individual target host nodes to each one. Click the **Edit** button associated with a farm entry in the Replication Farms window to display the Replication Farm window for that particular replication farm (Figure 60).

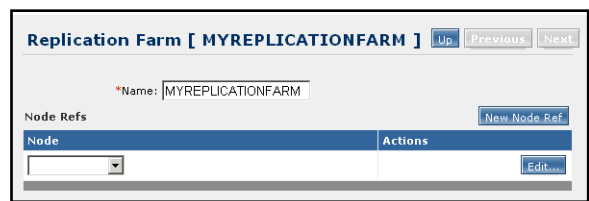


Figure 60: Replication Farm Window

Here you can select those target hosts listed in your nodes configuration file (by default `odnodes.xml`) from the **Node** list. If you want to assign more than one target node to the replication farm element, click **New Node Ref** to add another entry.

## Assigning Next-Tier Deployments to Target Hosts

If you intend for one or more of your deployment target hosts to redeploy the files it receives, you can configure that target host to trigger a specific deployment of its own upon receipt of the initial deployed files. This is part of a next-tier deployment. Any target you assign a next-tier deployment must have the base server software installed, and be able to deploy files to targets of its own.

Click the **Edit** button associated with your node entry in the Farm window to display the Node Ref window (Figure 61).



Deployment	Invoke On Success	Actions
<input type="text"/>	<input type="radio"/> yes <input checked="" type="radio"/> no	<input type="button" value="Delete"/>

Figure 61: Node Ref Window

Enter the name of the deployment you want the associated node to run following receipt of the initial deployment in the **Deployment** box. The deployment you enter must be present in the target host.

If you select the **yes** option under **Invoke On Success**, then the next-tier deployment will only be run if the target host receives the first deployment successfully.

Click **New Next Deployment** if you want to assign another next-tier deployment to your target host. You can also select a different target host from the **Node** list to which you can associate your next-tier deployment.

Specifying a Transactional Deployment.

A transactional deployment will automatically roll back a deployment and restore the target host’s files to their previous states in the event the deployment is not completely successful. This feature is particularly useful if you are deploying to multiple targets simultaneously, and it is important that all the target hosts contain the exact same files. See “Transactional Deployments” on page 229 for more information.

Select **Deployment** from the tree to display the Deployment window (Figure 62).

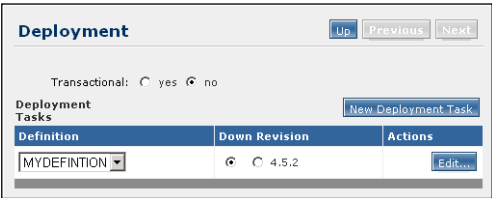


Figure 62: Deployment Window

To make your deployment transactional, click the **yes** button associated with the **Transactional** feature in the Deployment window.

Enabling Deployments to Downward Revision Hosts

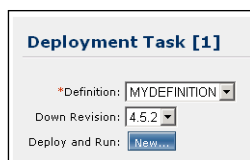
OpenDeploy 5.5.1 can deploy files to a target host running OpenDeploy 5.0.1 with no further modifications to the source or targets hosts required. However, deploying to a target host running OpenDeploy 4.5.2 requires both the OpenDeploy 4.5.2 sending (client) software be installed on the source host and the base server configuration file (by default `odbase.xml`) allow these types of down-rev deployments. See “Deploying to OpenDeploy Downward Revision Targets” on page 89 for more information.

The deployment configuration must also be modified accordingly to allow deployments to target hosts running OpenDeploy 4.5.2. You can configure OpenDeploy to deploy your configuration’s files to target hosts running the OpenDeploy 4.5.2 receiver software by clicking the button next to **4.5.2** in the Deployment window (Figure 62).

## Setting Deploy and Run

Deploy and Run is an OpenDeploy feature that allows you to run an external script during a deployment when one or more triggers apply. These triggers can include when a certain individual or category of directories or files are deployed, or when a particular deployment itself is run. Deploy and Run scripts associated with these triggers can be set to run on either the source host running the deployment, the host receiving the deployed files. They can also be set to trigger on the success of a deployment, its failure, or both. See “Deploy and Run” on page 267 for more information on this feature.

You can configure Deploy and Run in the Deployment Configuration Composer by clicking the **Edit** button associated with a definition element in the Deployment window (Figure 62). Clicking **Edit** will open the Deployment Task window (Figure 63).



**Deployment Task [1]**

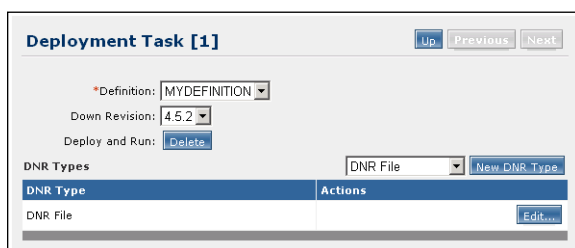
\*Definition: MYDEFINITION ▼

Down Revision: 4.5.2 ▼

Deploy and Run: [New...](#)

Figure 63: Deployment Task Window

Select the definition element to which you want to associate the Deploy and Run functionality by selecting its name from the **Definition** list. Click **New** to display additional Deploy and Run attributes in the Deployment Task window (Figure 64).



**Deployment Task [1]** [Up](#) [Previous](#) [Next](#)

\*Definition: MYDEFINITION ▼

Down Revision: 4.5.2 ▼

Deploy and Run: [Delete](#)

DNR Types DNR File ▼ [New DNR Type](#)

DNR Type	Actions
DNR File	<a href="#">Edit...</a>

Figure 64: Deployment Task Window with Deploy and Run Attributes

The Deploy and Run feature includes the following options:

- **DNR File** — select to specify under what conditions deployed files can trigger a Deploy and Run script.
- **DNR Directory** — select to specify under what conditions deployed directories can trigger a Deploy and Run script.
- **DNR Deployment** — select to specify under what conditions the running of a particular deployment can trigger a Deploy and Run script.

Select the appropriate choice from the **Type** list and click **New DNR Type** to add an entry for that Deploy and Run element. You can create elements for DNR File, DNR Directory, DNR Deployment, in any number and combination (Figure 65).

Deployment Task [1]

Up

Previous

Next

\*Definition:

MYDEFINITION

Down Revision:

4.5.2

Deploy and Run:

Delete

DNR Types

DNR Deployment

New DNR Type

DNR Type	Actions
DNR File	<div>Edit...</div> <div>Delete</div>
DNR Directory	<div>Edit...</div> <div>Delete</div>
DNR Deployment	<div>Edit...</div> <div>Delete</div>

Figure 65: Deployment Task Window with Multiple Deploy and Run Elements

## DNR File

Clicking the **Edit** button associated with a DNR File entry displays the DNR File window (Figure 66).

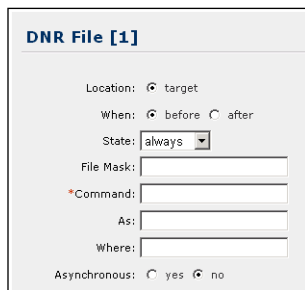


Figure 66: DNR File Window

The DNR File window contains the following attributes:

- **Location** option — this option is fixed as **target**, indicating the script will take place on the target host.
- **When** option — select whether the script should be executed **before** or **after** the deployment of the particular file.
- **State** list — indicates whether the Deploy and Run script should run as a result of the **success** or **failure** of the deployment, or whether it should **always** run in either case.

If the **When** option is set to **before**, then the **State** attribute is implicitly set to **always**, because there has been no deployment yet to determine success or failure.

- **File Mask** box — enter the regular expression specifying the deployed files that will trigger the script. If you entered the following value:

```
\.html$
```

any deployed file with the file extension `.html` will trigger the Deploy and Run script.

- **Command** box — enter the command where OpenDeploy can start a Deploy and Run script, as well as any accompanying flags or options. You can also specify an executable invocation line. For example:

```
C:\bin\email_to_admin.bat -user jdoe@interwoven.com or
/bin/mail jdoe@interwoven.com < /tmp/message.txt
```

If the command you are going to run requires a scripting engine, the scripting engine must be on the PATH of the user (or system, on Windows) who will be running the script or specified with a full path). For example:

```
/bin/sh /usr/local/bin/email_to_admin.sh -u jdoe@interwoven.com or
/usr/local/bin/iwperl /path/to/script.pl
```

- **As** box — enter a different user name or user ID under which you can run the script. If you entered the following value:

```
rroe or
100
```

you could run the script as rroe rather than as your regular user name. By default, the script runs as the user who invokes OpenDeploy, who will need to be root for most purposes.

This feature applies to UNIX hosts only. On Windows hosts, this feature always runs as *Default User*.

- **Where** box — enter the path to the location where the **Command** attribute value is to be run. For example:

```
/tmp or
C:\temp
```

You must use the path syntax supported by your OpenDeploy server. Forward slashes (“/”) can be used for either Windows or UNIX hosts, while backslashes (“\”) are only permitted on Windows hosts.

The **Where** attribute is optional. If you do not specify a value, the process takes place in the root directory.

- **Asynchronous** option — select **yes** to run the script asynchronously or **no** not to. Exercise caution when using this mode, as it could cause many scripts to be run simultaneously. The output from scripts run asynchronously is not captured.

## DNR Directory

Clicking the **Edit** button associated with a DNR Directory entry displays the DNR Directory window (Figure 67).



**DNR Directory [2]**

Location: ☒ target

When: ☒ before ☐ after

State:

Directory Mask:

\*Command:

As:

Where:

Asynchronous: ☐ yes ☒ no

Figure 67: DNR Directory Window

The DNR Directory window contains the following attributes:

- **Location** option — this option is fixed as **target**, indicating the script will take place on the target host.
- **When** option — select whether the script should be executed **before** or **after** the deployment of the particular directory.
- **State** list — select whether the Deploy and Run script should run as a result of the **success** or **failure** of the deployment, or whether it should **always** run in either case.

If the **When** option is set to **before**, then the **State** attribute is implicitly set to **always**, because there has been no deployment yet to determine success or failure.

- **Directory Mask** box — enter the regular expression specifying the deployed directories that will trigger the script. If you entered the following value:

```
cgi-bin$
```

any deployed directory in the deployment path named `cgi-bin` will trigger the Deploy and Run script.

- **Command** box — enter the command where OpenDeploy can start a Deploy and Run script, as well as any accompanying flags or options. You can also specify an executable invocation line. For example:

```
C:\bin\email_to_admin.bat -user jdoe@interwoven.com or
/bin/mail jdoe@interwoven.com < /tmp/message.txt
```

If the command you are going to run requires a scripting engine, the scripting engine must be on the PATH of the user (or system, on Windows) who will be running the script or specified with a full path). For example:

```
/bin/sh /usr/local/bin/email_to_admin.sh -u jdoe@interwoven.com or
/usr/local/bin/iwperl /path/to/script.pl
```

- **As** box — enter a different user name or user ID under which you can run the script. If you entered the following value:

```
rroe or
100
```

you could run the script as `rroe` rather than as your regular user name. By default, the script runs as the user who invokes OpenDeploy, who will need to be root for most purposes.

This feature applies to UNIX hosts only. On Windows hosts, this feature always runs as *Default User*.

- **Where** box — enter the path to the location where the **Command** attribute value is to be run. For example:

/tmp *or*

C:\temp

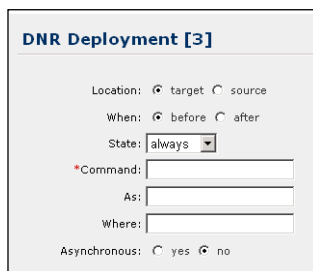
You must use the path syntax supported by your OpenDeploy server. Forward slashes (“/”) can be used for either Windows or UNIX hosts, while backslashes (“\”) are only permitted on Windows hosts.

The **Where** attribute is optional. If you do not specify a value, the process takes place in the root directory.

- **Asynchronous** option — select **yes** to run the script asynchronously or **no** not to. Exercise caution when using this mode, as it could cause many scripts to be run simultaneously. The output from scripts run asynchronously is not captured.

## DNR Deployment

Clicking the **Edit** button associated with a DNR Deployment entry displays the DNR Deployment window (Figure 68).



**DNR Deployment [3]**

Location: ☒ target ☐ source

When: ☒ before ☐ after

State:

\*Command:

As:

Where:

Asynchronous: ☐ yes ☒ no

Figure 68: DNR Deployment Window

The DNR Deployment window contains the following attributes:

- **Location** option — select whether the Deploy and Run script is taking place on the **source** or **target** host.
- **When** option — select whether the script should be executed **before** or **after** the deployment of the particular file.
- **State** list — indicates whether the Deploy and Run script should run as a result of the **success** or **failure** of the deployment, or whether it should **always** run in either case.

If the **When** option is set to **before**, then the **State** attribute is implicitly set to **always**, because there has been no deployment yet to determine success or failure.

- **Command** box — enter the command where OpenDeploy can start a Deploy and Run script, as well as any accompanying flags or options. You can also specify an executable invocation line. For example:

```
C:\bin\email_to_admin.bat -user jdoe@interwoven.com or
/bin/mail jdoe@interwoven.com < /tmp/message.txt
```

If the command you are going to run requires a scripting engine, the scripting engine must be on the PATH of the user (or system, on Windows) who will be running the script or specified with a full path). For example:

```
/bin/sh /usr/local/bin/email_to_admin.sh -u jdoe@interwoven.com or
/usr/local/bin/iwperl /path/to/script.pl
```

- **As** box — enter a different user name or user ID under which you can run the script. If you entered the following value:

```
rroe or
100
```

you could run the script as rroe rather than as your regular user name. By default, the script runs as the user who invokes OpenDeploy, who will need to be root for most purposes.

This feature applies to UNIX hosts only. On Windows hosts, this feature always runs as *Default User*.



- **Where** box — enter the path to the location where the **Command** attribute value is to be run. For example:

`/tmp or`

`C:\temp`

You must use the path syntax supported by your OpenDeploy server. Forward slashes (“/”) can be used for either Windows or UNIX hosts, while backslashes (“\”) are only permitted on Windows hosts.

The **Where** attribute is optional. If you do not specify a value, the process takes place in the root directory.

- **Asynchronous** option — select **yes** to run the script asynchronously or **no** not to. Exercise caution when using this mode, as it could cause many scripts to be run simultaneously. The output from scripts run asynchronously is not captured.

### Deleting Deploy and Run Entries

Each Deploy and Run element has associated attributes that you can display the complete by clicking the **Edit** button associated with the specific Deploy and Run element. If your deployment already has a Deploy and Run element configured, a **Delete** button will be displayed in place of the **New** button. Click this **Delete** button to delete all of the Deploy and Run elements you have created for the associated definition. If you want to delete specific Deploy and Run elements, but leave other intact, click the **Delete** button of the associated Deploy and Run element entry.

## Naming and Adding Definitions

A definition element in a deployment configuration specifies each pairing of one or more file system locations or TeamSite areas residing on the source host with a single target location that will receive the deployed files. In the case of file system comparison deployments, the files residing in the target location will also be compared with those in the source host file system location. You can have more than one definition element in a deployment configuration, and each one must have a unique name.

Enter a unique name for your definition in the **Definition** box in the Deployment Configuration window (Figure 54), for example:

MYDEFINITION

If you want more than one definition in your deployment configuration, click **New Definition** to add another **Definition** entry. You must provide a unique name for each definition in your configuration.

## Selecting the Definition Type

Each definition within a deployment can be one of the following types:

- **Forward Definition** — the deployment originates at source host (usually a development server) and sends files to a receiving host (usually a production server). This type of deployment definition follows the standard development workflow.
- **Reverse Definition** — the deployment originates at a receiving host (usually a production server) and deploys files back to the source host (usually a development server). This type of deployment definition will resynchronize the files between a development server and a production server if the production server has its files modified outside the standard workflow. See “Reverse Deployments” on page 236 for more information

You can select the definition type in the Definition window (Figure 69).

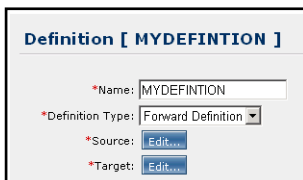


Figure 69: Definition Window

Select **Definition** from the tree to open this window. You can also click the **Edit** button associated with your definition entry in the Deployment Configuration window (Figure 62).

Select the type of definition you want from the **Definition Type** list.

## Defining the Source File Location

The source file location is where the deployable files reside on the source host. You can configure your source file location in the Source window. Select **Source** from the tree to display the Source window. You also can click the **Edit** button associated with the source in the Definition window (Figure 69). The contents of the Source window is determined by the type of deployment you are configuring.

### Directory Comparison Deployments

For directory comparison deployments, the Source window (Figure 70) displays attributes for the source file location and for overriding the target file location. The file list attribute only applies to file list deployments described later in this section. This is the default Source window displayed when you first access the Source window.

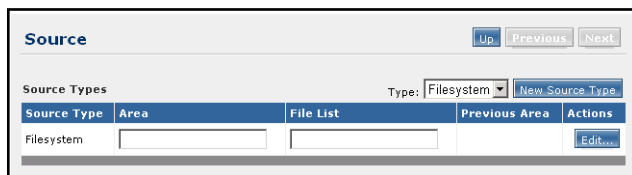


Figure 70: Source Window for Directory Comparison Deployments

Enter the absolute path to the file system location where your deployment's source files reside in the **Area** box. For example:

`/website/files` or

`C:\website\files`

The files in that location are compared with files residing on the target host, and the differences are deployed. See "Directory Comparison Deployments" on page 211 for more information.

TeamSite Comparison Deployments

For TeamSite comparison deployments, the Source window (Figure 71) displays attributes for the TeamSite area where the source files are located, and an alternate TeamSite area against which the first area will be compared and the differences deployed.

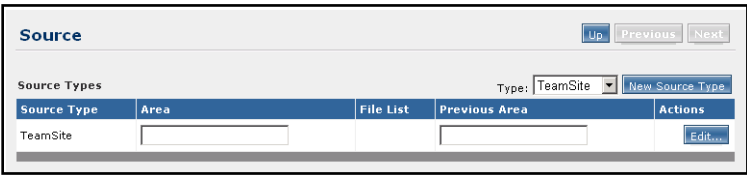


Figure 71: Source Window for TeamSite Comparison Deployments

By default, the Source window displays the attributes for a directory comparison deployment. To configure a TeamSite comparison deployment, you must change the Source window to display attributes for a TeamSite comparison deployment by selecting TeamSite from the **Type** list and clicking **New Source Type**. A source entry displaying the **Area** and **Previous Area** boxes is displayed. If you do not plan to use the existing **Filesystem** entry, remove it from the Source window by clicking its associated **Delete** button.

To configure your source file location information for a TeamSite comparison deployment, follow these steps:

1. Enter the full path to the TeamSite area where your deployment’s source files reside in the **Area** box. This TeamSite area can be a staging area, edition, or workarea. For example:

`//IWSERVER/default/main/dev/EDITION and`

2. Enter the full path to another TeamSite area on the source host against which the files residing in the **Area** box’s location will be compared and the differences deployed. For example:

`//IWSERVER/default/main/dev/EDITION/IW_PREV`

See “TeamSite Comparison Deployments” on page 214 for more information.

## File List Deployments

File list deployments are configured in the same Source window as is used for directory comparison deployments (Figure 70). File list deployments do not compare files. Instead, the files listed in a file list are deployed to the target. A file list deployment is configured similarly to a directory comparison deployment, except that the path to the file list on the source host is specified. The target host simply receives the files listed in the file list. See “File List Deployments” on page 219 for more information.

To configure your source file location information for a file list deployment, follow these steps:

1. Enter the absolute path to the file system location where your deployment’s source files reside in the **Area** box. For example:

```
/website/files or
```

```
C:\website\files
```

2. Enter the absolute path to the file list in the **Filelist** box of the **Filesystem** entry. For example:

```
C:\OpenDeploy\files\filelist.txt
```

## Editing Source File Configurations

Each source file entry you create will have an entry in the tree. Select that entry from the tree, or click the **Edit** button associated with that entry in the Source window to display the corresponding source entry window.

Figure 72 displays a sample of a source file entry for a directory comparison.

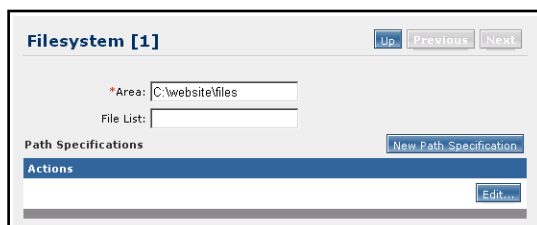


Figure 72: Filesystem Window for Editing a Directory Comparison Source File Entry

Figure 73 displays a sample of a source file entry for a TeamSite comparison.

Figure 73: TeamSite Window for Editing a TeamSite Comparison Source File Entry

Figure 74 displays a sample of a source file entries for a file list deployments.

Figure 74: Filesystem Window for Editing a File List Source File Entry

## Configuring Multiple Source Entries

You can configure more than one source entry within a definition. However, by default, all source entries deploy to the same target location. You run the risk of overwriting your files if you have multiple sources deploying simultaneously to the same target location. Multiple sources can also cause problems if one or more have the `doDeletes` feature enabled. In most cases, you only want to configure a single source entry. The one exception is when you use the Target Rules feature to define a different target location. See “Defining Source-Based Overrides” on page 223 for more information.

## Defining a Subdirectory Within the Source File Location

In some cases, you might want to further narrow the source file location for your deployment within the specified source file system location or TeamSite area. You can specify a subdirectory relative to the source file area by entering that directory path in the Path Specification window (Figure 75).



Figure 75: Path Specification Window

Select the **Path Specification** entry associated with your **Source** entry in the tree to display the corresponding Path Specification window. You can also click the **Edit** button in the corresponding source file entry (Figure 72).

Click **Path** to display the Path window ().

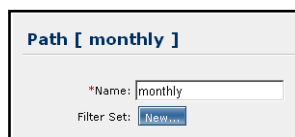


Figure 76: Path Window

Enter the relative path within the specified file system location or TeamSite area. For example, if you wanted to deploy files from the directory **monthly** within the specified area, you would enter the following value in the **Name** box:

**monthly**

If you do not want to specify a subdirectory within your specified area, simply enter a period (".") in the **Name** box.

You can specify additional subdirectories within your source file area by clicking the **New Path Specification** button in your file system or TeamSite window. Each path specification entry you add will have a corresponding path element that you can access by clicking the corresponding **Edit** button.

Each path entry you add is displayed in the tree as a separate **Path** entry. Select the **Path** entry or click the **Edit** button associated with your path entry in the Path Specification window to display the Path window. You can apply filters that will exclude specified files and directories from deployments at the target (see next section).

Applying Source-Side Filters

You can include filters that will exclude files and directories at the source host, based on patterns in their paths names or their file names. These filters are configured in the Path Specification window when the you click the **New** button associated with **Filter Set** (Figure 77).

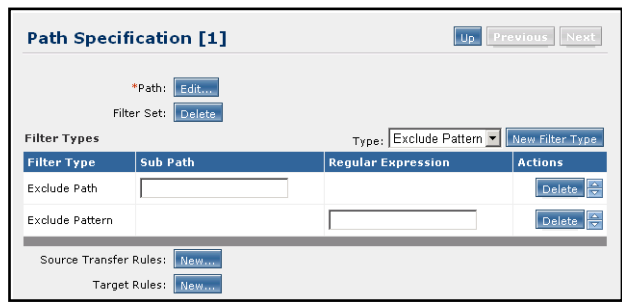


Figure 77: Path Specification Window with Filtering Attributes Displayed

You can display the Filter Set attributes by clicking the associated **New** button in the Path Specification window (Figure 75). If configured Filter Set attributes already exist, they will automatically be displayed in the Path Specification window. You can also select **Filter Set** from the tree if an entry already exists.

You can configure one or more entries each for the following filters:

- Exclude Path — ignores directories located in a specified path on the source host.
- Exclude Pattern — ignores files based on a specified regular expression on the source host during deployments.

See “Filtered Deployments” on page 249 for more information on these features.

Select the filter you want from the **Type** list and click **New Filter Type**.

### **Exclude Path**

The path you enter in the **Sub Path** box is relative to the local directory or TeamSite area on the source host containing the files to be moved. For example, if you wanted to exclude the directory `monthly`, which resides within the specified area, then you would enter the following value:

```
monthly
```

### **Exclude Pattern**

The regular expression you enter into the **Regular Expression** box determines which files will be ignored on the source host. For example, if you wanted to ignore any file that ends with the extension `.html`, then you would enter the following value:

```
\.html$
```

If you are not familiar with regular expressions, consult a reference manual such as *Mastering Regular Expressions* by Jeffrey Friedl.

### **Deleting Filters**

You can delete individual Exclude Path or Exclude Pattern entries by clicking the **Delete** button associated with the particular entry.

You can delete your entire set of filters by clicking the **Delete** button associated with the Filter Set feature in the Path Specification window.

### **Path Element Filters**

The Path window (Figure 76) also contains a button for applying filters to files associated with that path. The functioning of filters in the path element is the same as that for the path specification element already described.

## Following Source-Side Symbolic Links in Deployments

You can configure a deployment running on a UNIX server to deploy files referenced in symbolic links, a procedure known as *following links*. This feature is not available with OpenDeploy running on Windows. See “Deploying Symbolic Links” on page 264 for more information on this feature.

You can configure your UNIX OpenDeploy host to follow links on the source side by enabling the feature in the Source Transfer Rules window (Figure 78).

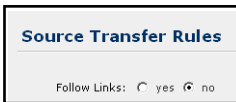


Figure 78: Source Transfer Rules Window

You can display the Source Transfer Rules window by clicking the associated **New** or **Edit** button in the Path Specification window (Figure 75). You can also select **Source Transfer Rules** from the tree if an entry already exists.

Select the **yes** option in the Source Transfer Rules window to enable the Follow Links feature.

You can also enable the follow links feature at the target side by enabling the Follow Links attribute in the Transfer Rules window. See “Applying Transfer Rules” on page 328 for more information.

## Designating Alternate Targets from the Source

You can associate a target area with a particular source in a file system comparison deployment with multiple sources. This feature provides the ability to create multiple source/target pairings for a single deployment. See “Defining Source-Based Overrides” on page 223 for more information on this feature.

You can designate an alternate target location for a set of deployed files through the Target Rules window (Figure 79).

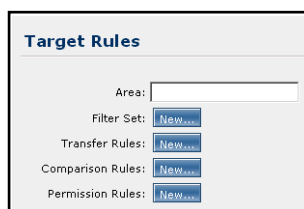


Figure 79: Target Rules Window

You can display the Target Rules window by clicking the associated **New** or **Edit** button in the Path Specification window (Figure 75). You can also select **Target Rules** from the tree if an entry already exists.

The Target Rules window contains the **Area** box, where you can enter the path for an alternate target file location. There are also buttons for other features that apply only to those files that are being deployed. Each of these features is described separately.

Enter the full path to the alternate target location for the source files that use this feature. For example, if you wanted to deploy to the following location:

D:\metadata\files

enter that path in the **Area** box.

Other features accessible in the Target Rules window that you can apply to alternate targets are listed below:

- **Filter Set** — see “Applying Target-Side Filters” on page 334.
- **Transfer Rules** — see “Applying Transfer Rules” on page 328.
- **Comparison Rules** — see “Applying Comparison Rules” on page 326.
- **Permission Rules** — see “Applying Permission Rules” on page 329.

## Defining the Target File Location

The target file location is the file system location on the target host where deployed files reside following the deployment. In directory comparison deployments, any files residing in the target directory location are compared with equivalent files in the source file location, and the differences (based on the criteria for deployment) are deployed to the target. In TeamSite comparison and file list deployments, the target file location is simply a repository for the deployed files. Any files residing in the target file location are excluded from the comparison of TeamSite areas on the source host. Files residing in the target file area prior to running the deployment can be overwritten by like-named deployed files.

By default, one target directory location is associated with each definition element in a deployment configuration. If you have multiple sources within a definition, they will (by default) deploy files to the same target directory location. You can use the target rules option to create alternate target directory locations on a source-by-source basis. See “Designating Alternate Targets from the Source” on page 324 for more information on that feature.

The target file location is defined in the Target window (Figure 80).

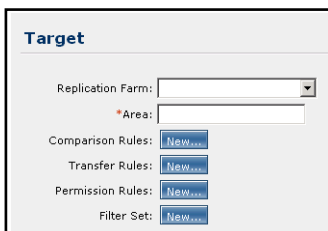


Figure 80: Target Window

You can display the Target window by clicking the associated **Edit** button in the Definition window (Figure 69), or by selecting the **Target** entry in the tree.

Each target must be associated with a particular replication farm element. The replication farm contains one or more individual target host nodes that will receive the deployed files. The information you enter into the Target window determines the target file location on the target host nodes making up the members of the replication farm. If you have more than one replication farm element defined in your configuration, select the appropriate choice from the **Replication Farm** list.

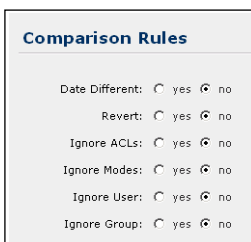
Enter the full path to the target file location in the **Area** box. This value must be a file system location. You cannot enter a TeamSite area. If you want to deploy files into a TeamSite workarea, enter that location as a file system path, for example:

```
Y:\default\main\dev\WORKAREA\jdoe
```

## Applying Comparison Rules

Comparison rules define the rules that OpenDeploy uses when it compares files contained in a file system. The type and platform of the file system determines the criteria available. These rules are used to determine eligibility of files for deployment. See “File Comparison Rules” on page 254 for more information on this feature.

Comparison rules are defined in the Comparison Rules window (Figure 81).



The screenshot shows a window titled "Comparison Rules" with a list of settings, each with a radio button for "yes" and "no". The "no" option is selected for all settings.

Comparison Rules	
Date Different:	<input type="radio"/> yes <input checked="" type="radio"/> no
Revert:	<input type="radio"/> yes <input checked="" type="radio"/> no
Ignore ACLs:	<input type="radio"/> yes <input checked="" type="radio"/> no
Ignore Modes:	<input type="radio"/> yes <input checked="" type="radio"/> no
Ignore User:	<input type="radio"/> yes <input checked="" type="radio"/> no
Ignore Group:	<input type="radio"/> yes <input checked="" type="radio"/> no

Figure 81: Comparison Rules Window

You can display the Comparison Rules window by clicking the **New** or **Edit** button associated with Comparison Rules in the Target window (Figure 80), or by selecting its entry from the tree.

The Comparison Rules window contains the following options:

- **Date Different** option — indicate whether or not a file should be deployed if there is any difference in file date (older or newer) between the source and target versions. This differs from the OpenDeploy default date-based comparison setting, where a file should be deployed only if the source file is newer than the target file.
- **Revert** option — indicate whether or not a file should be deployed if the source version is older than the target version.
- **Ignore ACLs** option (Windows only) — indicate whether (**yes**) or not (**no**) to ignore differences in the Windows access control lists (ACLs) during the file comparison.
- **Ignore Modes** option (UNIX only) — indicate whether (**yes**) or not (**no**) to ignore differences in the UNIX-based permission mode bits definitions during the file comparison.
- **Ignore User** option (UNIX only) — indicate whether (**yes**) or not (**no**) to ignore differences in the UNIX-based user ownership during the file comparison.
- **Ignore Groups** (UNIX only) option — indicate whether (**yes**) or not (**no**) to ignore differences in the UNIX-based group ownership during the file comparison.

## Applying Transfer Rules

Transfer rules define the rules for moving files from the source host to the target host during the deployment. See “File Transfer Rules” on page 256 for more information on this feature.

Transfer rules are defined in the Transfer Rules window (Figure 82).

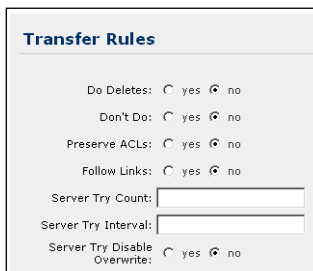


Figure 82: Transfer Rules Window

You can display the Transfer Rules window by clicking the **New** or **Edit** button associated with Transfer Rules in the Target window (Figure 80), or by selecting its entry from the tree.

The Transfer Rules window contains the following options:

- **Do Deletes** option — select whether (**yes**) or not (**no**) files and directories not present in the source host area will be deleted on the target host.
- **Don't Do** option — select whether (**yes**) or not (**no**) to proceed with the deployment following the comparison. Deployment will not occur if this attribute is enabled. This is a good tool to use to check and compare files without actually performing a deployment.
- **Preserve ACLs** option (Windows only) — select whether (**yes**) or not (**no**) to preserve the Windows access control lists (ACLs) when the files are moved.
- **Follow Links** option — select whether (**yes**) or not (**no**) symbolic links on the target hosts will be followed when the files are moved.
- **Server Try Count** box (Windows only) — enter the number of times OpenDeploy will attempt to deploy the file to the target host. This feature works in conjunction with Microsoft IIS, and is designed to accommodate times of heavy production server traffic.

- **Server Try Interval** box (Windows only) — enter the amount of time in seconds OpenDeploy waits between deployment attempts. This feature works in conjunction with Microsoft IIS, and is designed to accommodate times of heavy production server traffic.
- **Server Try Disable Overwrite** option (Windows only) — select whether (**yes**) or not (**no**) to disable the ability of OpenDeploy to deploy files to a server even if the `svrTryCount` and `svrTryInterval` elements are specified. This feature works in conjunction with Microsoft IIS, and is designed to accommodate times of heavy production server traffic.

Applying Permission Rules

Permission rules define the rules applicable to the permissions of deployed files and directories. See “File Permission Rules” on page 258 for more information on this feature.

Permission rules are defined in the Permission Rules window (Figure 83).

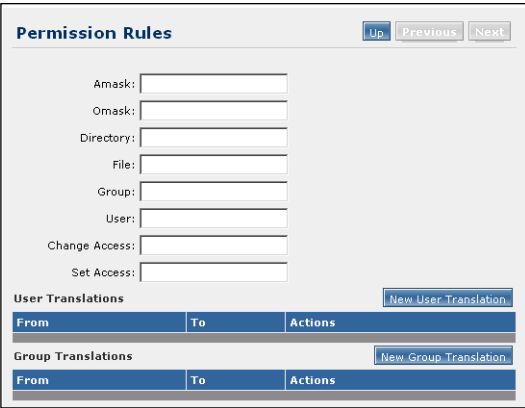


Figure 83: Permission Rules Window

You can display the Permission Rules window by clicking the **New** or **Edit** button associated with Permission Rules in the Target window (Figure 80), or by selecting its entry from the tree.

Access options specific to UNIX are ignored when deploying to a Windows target host, and access options specific to Windows are ignored when deploying to a UNIX target host.

The Permission Rules window contains the following options:

- **Amask** box (UNIX only) — enter the bit mask (in octal) to be ANDed with the permission bits of all files and directories. The amask octal value combines with the existing permission bit value of the affected file. If a file has the existing permission value of 664 (-rw-rw-r--) and the amask attribute as the following value:

```
amask="770"
```

then the resulting permission for that file (664 AND 770) following the deployment would be 660 (-rw-rw----).

- **Omask** box (UNIX only) — enter the bit mask (in octal) to be ORed with the permission bits of all files and directories. The omask octal value combines with the existing permission bit value of the affected file. If a file has the existing permission value of 666 (-rw-rw-rw-) and the omask attribute as the following value:

```
omask="022"
```

then the resulting permission for that file (666 OR 022) following the deployment would be 644 (-rw-r--r--).

- **Directory** box (UNIX only) — enter the permissions (in octal) given to all deployed directories. For example, if you wanted deployed directories to have the permission “drwxrwx---”, then the resulting value would be:

```
directory="770"
```

- **File** box (UNIX only) — enter the permissions (in octal) given to all deployed files. For example, if you wanted deployed files to have the permission “-rw-rw-r-x”, then the resulting value would be:

```
file="665"
```

- **Group** box (UNIX only) — enter the group assigned to all deployed files and directories. This attribute value must be a valid group name or group ID. For example:

```
group="tech_pubs" or
```

```
group="200"
```

You must also specify the `user` attribute if you use employ the `group` attribute.

- **User** box (UNIX only) — enter the user who will own all deployed files and directories. This attribute value must be a valid user name or user ID. For example:

```
jdoe or
```

```
105
```

You must also specify the `group` attribute if you use employ the `user` attribute.

- **Change Access** box (Windows only) — enter a value that modifies the access control lists (ACLs) so that specified users have the designated rights. The new access control entry (ACE) for each specified user allows only the specified rights, discarding any existing ACE. In the following example:

```
changeAccess="{ jdoe:W, tech_pubs:NONE }"
```

any existing ACEs for `jdoe` and `tech_pubs` are removed, `jdoe` is granted write access, and the group `tech_pubs` has no access at all. Any other access rights that may have existed for other users are left unchanged. See “Using OpenDeploy with ACLs” on page 262 for more information.

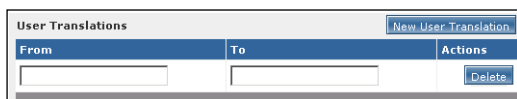
- **Set Access** box (Windows only) — enter a value that replaces the ACLs for the deployed files and directories. In the following example:

```
setAccess="{ jdoe:ALL, tech_pubs:RX }"
```

the existing ACL is removed and the user `jdoe` is granted full access. The group `tech_pubs` has read access to the specified files. Any other access rights that may have existed for the file are removed. See “Using OpenDeploy with ACLs” on page 262 for more information.

## User Translation

The User Translation feature (Figure 84) allows you to change user IDs (UNIX only) for deployed files. You can add and configure an unlimited number of user translations in your deployment. See “User and Group Ownership Transferral” on page 260 for more information on this feature.



User Translations		
From	To	Actions
jdoe or 105	rroe or 110	Delete

Figure 84: User Translation Feature

The User Translation feature contains the following attributes:

- **New User Translation** button — click to add a user translation entry.
- **From** box — enter the existing source user or user ID (the identification number assigned to each user account within the UNIX server). For example:

jdoe or

105

- **To** box — enter the new target user or user ID. For example:

rroe or

110

- **Delete** button — click to delete a user translation entry.

Group Translation

The Group Translation feature (Figure 85) is where you can change group IDs (UNIX only) for deployed files. You can add and configure an unlimited number of group translations in your deployment. See “User and Group Ownership Transferral” on page 260 for more information on this feature.

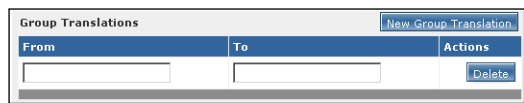


Figure 85: Group Translation Feature

The Group Translation feature contains the following attributes:

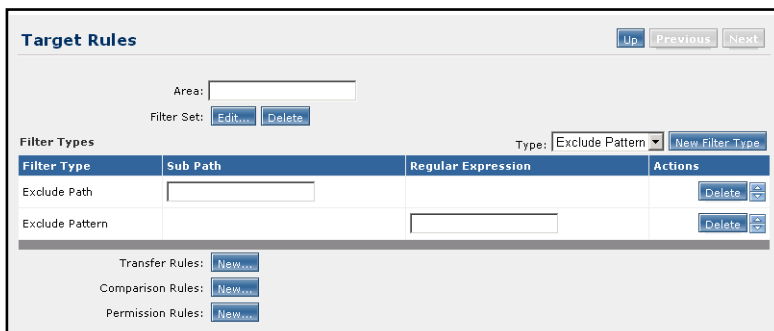
- **New Group Translation** button — click to add a group translation entry.
- **From** box — enter the existing source group or ID (the identification number assigned to each group account within the UNIX server). For example:  

tech\_pubs *or*  
100
- **To** box — enter the new target group or ID. For example:  

marketing *or*  
200
- **Delete** button — click to delete a group translation entry.

## Applying Target-Side Filters

You can include filters that will exclude files and directories at the source host, based on patterns in their paths names or their file names. Click **Filter Set** in the Target Rules window to display the filtering configuration information (Figure 86):



**Target Rules** [Up] [Previous] [Next]

Area:

Filter Set: [Edit...] [Delete]

Filter Types Type: Exclude Pattern [New Filter Type]

Filter Type	Sub Path	Regular Expression	Actions
Exclude Path	<input type="text"/>		[Delete] [v]
Exclude Pattern		<input type="text"/>	[Delete] [v]

Transfer Rules: [New...]  
 Comparison Rules: [New...]  
 Permission Rules: [New...]

Figure 86: Target Rules Window With Filtering Attributes Displayed

You can add as many Exclude Path and Exclude Pattern entries as you want by selecting the appropriate choice from the **Type** list and clicking **New Filter Type**.

### Exclude Path

The path you enter in the **Sub Path** box is relative to the local directory or TeamSite area on the source host containing the files to be moved. For example, if you wanted to exclude the directory `monthly`, which resides within the specified area, then you would enter the following value:

`monthly`

### Exclude Pattern

The regular expression you enter into the **Regular Expression** box determines which files will be ignored on the source host. For example, if you wanted ignore any file that ends with the extension `.html`, then you would enter the following value:

`\.html$`

If you are not familiar with regular expressions, consult a reference manual such as *Mastering Regular Expressions* by Jeffrey Friedl.

See “Filtered Deployments” on page 249 for more information on these features.

## Saving the Deployment

When you have completed configuring your deployment, you must save it by clicking the **Save** button at the top of the Deployment Configuration Composer window. If any required items of information are missing, such as required attribute values, the Deployment Configuration Composer will display the appropriate error messages in the Error pane. See “Details Pane” on page 295 for an example.

Upon successfully saving the deployment, the configuration is displayed (Figure 87).

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
- <deploymentConfiguration>
  <logRules maxBytes="32Mb" level="verbose" />
  <localNode host="JMOOREBW2K" />
  - <replicationFarmSet>
    - <replicationFarm name="MYFARMNAME">
      <nodeRef useNode="MyLocalHost" />
    </replicationFarm>
  </replicationFarmSet>
  - <definition name="MYDEFINITIONNAME">
    - <source>
      - <sourceFilesystem
        area="C:\Interwoven\OpenDeployNG\conf\dtd"
        fileList="">
        - <pathSpecification>
          <path name="*" />
        </pathSpecification>
      </sourceFilesystem>
    </source>
    - <target useReplicationFarm="MYFARMNAME">
      <targetFilesystem
        area="C:\Interwoven\OpenDeployNG\tmp" />
      <comparisonRules dateDifferent="yes" revert="no"
        ignoreAcls="no" ignoreModes="no" ignoreUser="no"
        ignoreGroup="no" />
      <permissionRules amask="" omask="" directory="0755"
        file="0644" group="" user="" changeAccess=""
        setAccess="" />
    </target>
  </definition>
  - <deployment transactional="no">
    <execDeploymentTask
      useDefinition="MYDEFINITIONNAME" downRev="" />
    </deployment>
  </deploymentConfiguration>
```

Figure 87: Example of Saved Deployment

## Editing Deployment Configurations

You can use the Deployment Configuration Composer to edit existing configurations, even those that were not created using this tool. You can edit the configuration of any deployment that resides in the *od-admin/conf* directory, and that conforms to XML-based structure used in OpenDeploy 5.x deployment configurations.

To edit an existing deployment configuration, follow these steps:

1. Select **Configurations > View Configurations** to display the Deployment Configuration window.
2. Select the OpenDeploy host on which the deployment configuration will reside from the **Selected Host** list.
3. Select the deployment you want to edit from the **Deployment** list.
4. Click **Edit** to open a new browser window containing the Deployment Configuration Composer. The elements and attributes for that deployment are displayed in the Details pane.
5. Modify your deployment using the methods described in this chapter.
6. Click **Save** to save your changes.

# Glossary

---

This glossary lists terms and their definitions found in this manual.

Administrator role	The highest level of OpenDeploy access. An individual with the Administrator role has the ability to configure OpenDeploy hosts, create and start deployment configurations, and set access restrictions for individuals in the User role.
administration server	A server with the OpenDeploy administration software installed. The administration server is responsible for generating and managing the browser-based user interface in the OpenDeploy environment.
area	A file system- or TeamSite-based location where files reside or will reside as the result of a deployment. Files residing in one area can also be compared with files in another area to determine whether they should be deployed.
asymmetric key encryption	A high level (168-bit) of file encryption you can assign to deployed files. Asymmetric key encryption requires setting up a Secure Sockets Layer (SSL) <i>certificate authority</i> and generating the <i>certificate</i> .
attribute	A directive with a corresponding value that can be used to alter the default behavior of OpenDeploy, or that must be used to provide required information.
base server	The OpenDeploy software installed on a server that is licensed to send and receive deployments.
base server configuration file	An XML-based file that defines the rules and settings for a base server within the OpenDeploy environment. The base server configuration file must follow the XML rules as defined in the deploy server configuration DTD.
base server log	A log file containing entries related to the base server host.
bind port number	The number for the port that source and target hosts use to transport deployed files.
bootstrap administrator	The initial Administrator role user identity used to assign the Administrator role to other individuals.



certificate	A file which assures that both the source and target hosts in an asymmetric key encrypted deployment are certified to be taking part.
certificate authority	A set of programs used to generate public and private key pairs, and a database that contains state information. Certificate authority is used in conjunction with <i>asymmetric key encryption</i> .
cipher	An encryption tool that the source and target hosts share to hide the identity of deployed files.
compare phase	The period of time during a deployment when files are being compared to determine whether they should be deployed. This is also the time during a deployment when it can be cancelled.
definition	The matching of one or more source file locations (either file system locations or TeamSite areas) with a target file location (a file system location) for the purpose of deploying and receiving files. A deployment configuration can have one or more definitions between file locations on the sending host and the target file location.
Deploy and Run	An OpenDeploy feature that allows external scripts to be integrated into the deployment process.
deploy server configuration DTD	The XML-based DTD upon which the base server and receiver configuration files is derived.
deployment	The moving of files from a source host to one or more target hosts based on a particular deployment configuration.
deployment cancellation window	The combination of the compare phase and the pre-commit phase (transactional deployments only) when cancellation of a deployment in progress is possible.
deployment configuration	A combination of elements and attribute values which define the criteria for if and how files are to be deployed.
deployment configuration file	An XML-based file that defines the rules and settings for a source host to deploy files to one or more target hosts. The deployment configuration file must follow the XML rules as defined in the deployment configuration DTD.
deployment-based Deploy and Run	A Deploy and Run script that triggers when a certain deployment is run.
deployment criteria	The conditions that indicate whether a file should be deployed as part of a deployment configuration based on particular criteria.

development server	A server within the organizational firewall where Web sites are developed and tested prior to being deployed to a production server.
directory comparison	A type of deployment configuration where an area on the source and target hosts are compared with each other, and the differences, based on a set of specified criteria, are deployed to the target host.
directory-based Deploy and Run	A Deploy and Run script that triggers when a certain type or class of directories are deployed.
edition	A TeamSite term for a snapshot of files residing in the staging area for the purposes of archiving.
element	A logical unit of information within an XML-based document.
encryption	The ability to obscure the content of deployments moving between the source and target hosts to prevent unauthorized access.
fan-out deployment	A deployment configuration where the source host simultaneously deploys files to multiple target hosts.
file list deployment	A deployment configuration where the source host deploys files to the target hosts based on a predetermined list of files.
file comparison rules	Optional criteria to use for determining whether to deploy files from the source host to the target host.
file permission rules	Optional directives to apply to the deployed files or directories on the target host.
file transfer rules	Optional directives related to moving files from the source host to the target hosts.
file-based Deploy and Run	A Deploy and Run script that triggers when a certain type or class of files are deployed.
filtering	Specification of directory paths or file name patterns to exclude from consideration during the comparison phase, or exclude from being deployed.
group translation	The switching of UNIX-based group ownership on a deployed file or directory between the source host and the target host.
host	A server on which one or more OpenDeploy software components reside.
log files	Files containing information related to the status of the OpenDeploy server or a particular deployment.

logging level	One of the choices that determines the amount and type of logged information regarding deployments.
macro deployment log	The log file containing entries related to the activities involving a deployment configuration. This includes all individual source/target host pairings if the deployment has multiple targets.
micro deployment log	The log file containing entries related to the activities involving each individual source/target pairing of a deployment.
multi-host installation	The installation of all the required OpenDeploy software components (base server, administration server, and operations server) on multiple servers in some combination.
multi-tiered deployment	A deployment configuration where deployed files are received by an OpenDeploy base server host and then redeployed across <i>tiers</i> .
named deployment	A particular deployment configuration identified by its unique name.
node	A host that can send or receive files.
node configuration file	An XML-based configuration file that defines all the target hosts for a source host.
normal logging level	A logging level that logs standard status and error messages.
operations server	A server on which operations server software is installed. Operations server software determines roles and permissions of administrators and users within the OpenDeploy environment.
parameter substitution	A special attribute value syntax that allows you to specify a particular value for that attribute when starting a deployment using the <code>iwodstart</code> command-line tool. Using this feature, you can specify different values for the same parameter each time you start a deployment.
path	An element that further defines a specified file system location or TeamSite area.
pre-commit phase	A period of time when a transactional deployment determines whether the deployment has been successful and can commit the deployment to all the targets. If the deployment cannot commit, the deployment is rolled back and the target hosts are restored to their previous states.
previous area	A second TeamSite area against which the primary area is compared in a TeamSite comparison deployment configuration. The previous area is typically the previous version of the files in the primary area, and also represents the files residing on the target hosts.

primary area	The TeamSite area in a TeamSite comparison that contains the most current files. The contents of the primary area are compared with those in the previous area to determine which files are deployable.
production server	Typically, a host running a Web server that is accessible either on an intranet or the Internet.
receiver	A host on which OpenDeploy receiver software is installed. A receiver can only receive deployed files from a source host.
receiver log	A log file containing entries related to the receiver host.
receiver macro deployment log	A log generated by the receiving host that contains information for the received deployment as a whole.
receiver micro deployment log	A log generated by the receiving host that contains information regarding a specific source/target pairing in a deployment.
recurring deployment	A deployment configuration where the deployment repeats on a regular basis, such as daily or weekly.
revert	File comparison attribute where a file is deployed if the source version is older than the target version.
reverse deployment	A deployment configuration where files residing on a target host are deployed back to the source host.
reverse source	The sender of a reverse deployment.
reverse target	The recipient of a reverse deployment.
RMI port number	The number of the port used by the base server (source host), administration server, and operations server to communicate with each other.
roles	The collective term for the Administrator and User roles. The role of an individual user determines what level of features and functionality that person has access to within the OpenDeploy user interface. See also <i>Administrator role</i> and <i>User role</i> .
rollover threshold	The size a log file can grow before it is closed to new entries and archived. A new log file is then generated.
schedule	A predetermined time and date when a particular deployment configuration is started. This can occur on a one-time only or <i>recurring</i> basis.
scheduled deployment	A deployment configuration that is performed at a predetermined time and date as set in the user interface. A scheduled deployment can occur on a one-time or <i>recurring</i> basis.

scheduler database	A database that is installed with the base server software and stores the scheduling information for the source host.
service configuration file	A configuration file located on an OpenDeploy host with the base server or receiver software installed that specifies the name of the base server, receiver, and nodes configuration file being accessed by OpenDeploy. The service configuration file is named <code>deploy.cfg</code> and is located in the <i>od-home/etc</i> directory.
simulated deployment	A deployment where no files are moved, but entries are made in the deployment logs for every file or directory that would have been deployed. This feature can be used to determine differences in files on the source and target hosts without actually deploying files.
single-host installation	The installation of all the required OpenDeploy software components (base server, administration server, and operations server) on a single server.
source host	A host with base server software installed and licensed that can deploy and receive files.
source macro deployment log	A log generated by the sending host that contains information for the sent deployment as a whole.
source micro deployment log	A log generated by the sending host that contains information on a specific deployment source/target pairing. If the deployment moves files to multiple target hosts, each source/target pairing will have its own micro deployment log.
staging area	A TeamSite area that receives and stores files submitted to it from the <i>workareas</i> it supports. There is a single staging area for each TeamSite branch.
submit	A TeamSite task action where files are moved from a workarea to the staging area.
symmetric key encryption	A lower level (40-bit) of encryption you can assign to deployed files.
target host	A host with either receiver or base server software installed, which can receive deployed files from a source host.
TeamSite	Content management software. OpenDeploy is designed to work closely with TeamSite, and to utilize its features and functionality.
TeamSite comparison	A deployment configuration where two TeamSite <i>areas</i> are compared on the source host, and the differences are deployed to the specified target hosts. The source host must be configured as a TeamSite server.

test deployment	A deployment configuration that comes with OpenDeploy. The test configuration uses default settings to send the file <code>oddeployment.dtd</code> to the <code>od-home/tmp</code> directory. Using this test determines whether your base server host is properly configured.
tier	A grouping of a source host and its target hosts, usually in the context of a <i>multi-tiered deployment</i> .
Tomcat server	Software included in the base server software which assists in the generation of the OpenDeploy user interface.
transactional deployment	A feature which restores one or more targets to their previous existing states in the event that the deployment is considered unsuccessful.
user interface	The browser-based graphical representation of selected OpenDeploy functions you can use as an alternative to modifying configuration files and entering commands at the command prompt. The user interface provides an easy way to perform certain (but not all) OpenDeploy tasks and configurations.
User role	A lower level of OpenDeploy access. Users can only start and monitor those deployment configurations assigned to them by the administrator.
user translation	The switching of UNIX-based user ownership on a deployed file or directory between the source host and the target host.
workarea	A TeamSite area where contributors keep their working files.
verbose logging level	The highest level of deployment logging; detailed entries are written to the logs as the deployment occurs. This is the default logging level.



# Index

---

- A**
- ACEs
  - perm bits 262
  - standard perms 263
  - types 262
- ACLs 255, 262
  - names 262
- administration server 25, 47, 117
  - installation 53
  - opendeploy.war file 48, 95
  - software 46
  - starting 112
  - stopping 114
  - Tomcat server 48
  - uninstallation 107
  - UNIX 47
  - Windows 47
- Administrator role 42, 132, 134, 136
- allowedDirectories element 84, 115
- allowedHosts element 83, 84, 115
- amask attribute 258
- area (sourceFilesystem)
  - attribute 212, 222
- area (sourceTeamsite)
  - attribute 214, 215, 217, 219, 291
- area (targetFilesystem)
  - attribute 213
- area (targetRules) attribute 223, 225
- areas
  - alternate target 223
  - filesystem-based 212, 213, 219
  - TeamSite area-based 215
- as attribute 274
- asymmetric key encryption 282
  - certificate authority 283, 284
  - certificate generation 285
  - down-rev deployments 288
  - prerequisite tasks 283
- async attribute 274
- attribute
  - keywords 198
  - syntax 197
  - types 197
- attributes
  - amask 258
  - area (sourceFilesystem) 212, 222
  - area (sourceTeamSite) 219
  - area (sourceTeamsite) 214, 215, 217, 291
  - area (targetFilesystem) 213
  - area (targetRules) 223, 225
  - as 274
  - async 274
  - bindPort 76
  - bufferSize 77
  - changeAccess 259
  - class 78
  - cmd 273
  - dateDifferent 254
  - dbPassword 78, 81
  - dbUrl 78, 79, 81
  - dbUser 78, 81
  - deployment 235
  - directory 86, 156, 172, 258
  - doDeletes 253, 256
  - dontDo 256
  - downRev 89, 210
  - file 258
  - filelist 219, 220, 221, 222
  - followLinks
    - (sourceTransferRules) 264
  - followLinks
    - (transferRules) 256, 264
  - from 260
  - group 259
  - host 72, 203
  - ignoreAcls 254
  - ignoreGroup 255
  - ignoreModes 254
  - ignoreUser 255
  - invokeOnSuccess 235
  - isClearPassword 78, 81
  - jdbcDriverClass 81
  - level 86, 172
  - location (dnrDeployment) 272
  - location (dnrDir) 270
  - location (dnrFile) 269



- mask (dnrDir) 271
- mask (dnrFile) 269
- maxBytes 86, 171, 174
- name (node) 72
- name (path) 84
- name (replicationFarm) 204
- name (sourceFilesystem) 212
- nodes 83
- omask 258
- path 89
- pendSessions 88
- port 72
- preserveAcls 256
- previousArea 153, 215, 216, 217, 219, 291
- regex 251
- revert 254
- rmReadOnly 257
- setAccess 259
- sslCaCertificate 287
- sslCertificate 287
- sslCiphers 288
- sslPrivateKey 287
- sslVerifyPeer 287
- state (dnrDeployment) 272
- state (dnrDir) 271
- state (dnrFile) 269
- subPath 249
- svrTryCount 257
- svrTryDisableOverwrite 257
- svrTryInterval 257
- to 261
- transactional 209
- useDefinition 232
- useNode 204, 231
- user 259
- useReplicationFarm
  - (reverseSource) 237

- useReplicationFarm
  - (target) 207
  - version 77
  - when (dnrDeployment) 272
  - when (dnrDir) 270
  - when (dnrFile) 269
  - where 273

## B

- base server 24, 48, 49
  - bootstrap administrator 69
  - firewall authentication 104
  - installation 53
  - logging 159, 176
  - modifying 65
  - software 46
  - starting 112
  - stopping 115
  - uninstallation 107
  - UNIX 49
  - Windows 49
- base server configuration
  - downward TeamSite releases 77
- base server configuration file 74, 128
  - allowed directories 84
  - allowed hosts 83
  - bind port 76
  - buffer size 77
  - encoding 76, 96
  - encryption 87
  - host communication 76
  - logging 85
  - scheduler database 78
  - uploading 127
- base server log file 126, 159
- recovery 176

- bind ports 76
- bindPort attribute 76
- bootstrap administrator 55, 57, 66, 121
  - base server 69
  - configuring 67
  - daemons 68
  - OpenAPI 68
  - operations server 68
  - services 68

- browsers
  - refresh requirements 119
- buffer size 77
- bufferSize attribute 77

## C

- certificate authority 284
- certificate generation
  - UNIX 285
- changeAccess attribute 259
- ciphers 288
  - default 290
  - high-strength 290
  - low-strength 289
  - no-authentication 289
  - types 289
- class attribute 78
- cmd attribute 273
- command-line tools
  - iwodservergetversion 130
  - iwodserverstatus 131
- comparison rules 326
- comparisonRules element 254, 255
- configuration files
  - base server 74, 127, 128
  - host 128
  - nodes 70, 128

- receiver 90, 127, 128
- service 65
- Tomcat 92, 93, 94
- uploading 127

## D

- daemons 111
  - resetting 115
  - stopping 106
- DataDeploy integration 97
- dateDifferent attribute 254
- dbPassword attribute 78, 81
- dbUrl attribute 78, 79, 81
- dbUser attribute 78, 81
- definition element 205, 229, 232
- definitions 30, 205, 209
  - file filters 208
  - file rules 208
  - source file location 206
  - target file location 207
  - types 315
- Deploy and Run 210, 267, 306
  - communicating status 278
  - configuration 268
  - deleting 314
  - deployment-based 272, 312
  - directory-based 270, 310
  - file-based 269, 308
  - log capturing 277
  - logging 275
  - package files 278
  - requirements 268
  - scripting 273
  - secure invocation 267
- deployment attribute 235
- Deployment Configuration
  - Composer 141, 293
  - accessing 293

- comparison rules 326
- definitions 297, 315
- Deploy and Run 306
- details 295
- editing deployments 336
- encryption 301
- errors tab 294
- file naming 298
- filters 321, 334
- follow links 323
- forward deployments 315
- global deployment settings 296
- group translation 333
- legacy OpenDeploy
  - targets 305
- log rules 299
- new configurations 298
- next-tier deployments 304
- permission rules 329
- replication farms 303
- reverse deployments 315
- saving 335
- settings 296
- source file location 316, 320
- source host name 300
- target file location 325
- target host nodes 303
- targets, alternate 324
- transactional deployments 305
- transmission rules 328
- tree tab 294
- user translation 332
- deployment configurations 28
  - ACLs 262
  - allowed directories 84, 92
  - allowed hosts 83, 91
  - comparison rules 326
  - composing 140, 141

- creating 298
- definitionexecDeploymentTask
  - element 209
- definitions 30, 205, 297, 315
- Deploy and Run 210, 267, 306
- deployment host 203
- deployment tasks 208
- downward revision 89, 210
- editing 336
- encoding 96, 198
- encryption 87, 92, 301
- file comparison rules 254
- file naming 199, 298
- file permission rules 258
- file transfer rules 256
- filters 249, 321, 334
- forward deployments 315
- group translation 333
- log rules 299
- logging 153, 228
- mixed target platforms 225
- next-tier 304
- overrides 223, 225
- parameter substitution 265
- permission rules 329
- replication farms 303
- reverse deployments 315
- samples 238
- saving 335
- settings 296
- source file location 206, 316, 320
- source host name 300
- source-based overrides 223
- structure 199
- symbolic links 264
- target file location 207, 325
- target nodes 303

- target replication farms 203
  - target-based overrides 225
  - targets, alternate 324
  - transactional 208, 305
  - transmission rules 328
  - uploading 192
  - user translation 332
  - XML code 147, 195
  - deployment criteria 31
    - comparison-based 31
    - filelist-based 32
  - deployment element 208, 229
  - deployment job queuing 88
  - deploymentConfiguration
    - element 199
  - deployments
    - authorization 138, 139
    - cancellation window 154
    - cancelling 153
    - compare phase 153
    - directory comparison 31, 34, 211
    - fan-out 34, 38, 230
    - file list 32, 35, 219, 223
    - filtered 249
    - forward 315
    - legacy OpenDeploy
      - targets 305
    - legacy Web sites 291
    - monitoring 148
    - multi-tiered 39, 233
    - package 278
    - pre-commit phase 154
    - reverse 41, 236, 315
    - running 141
    - scenarios 37
    - scheduled 177
    - scheduling 179
    - simulated 145
    - single target 37
    - starting 141, 142, 143, 185
    - TeamSite comparison 31, 34, 214, 215, 218
    - test 145
    - transactional 40, 229
    - types 31, 33, 195, 293
  - deployNRun element 210, 268
  - directory attribute 86, 156, 172, 258
  - directory comparison 31, 34, 211
    - area 212, 213
    - source host 212
    - target hosts 213
  - dnrDeployment element 269, 272
  - dnrDir element 269, 270
  - dnrFile element 269
  - documentation,
    - OpenDeploy 11, 13
  - doDeletes attribute 253, 256
  - dontDo attribute 256
  - downRev attribute 89, 210
  - downward revision
    - deployments 89, 210
- E**
- elements 195
    - allowedDirectories 84, 115
    - allowedHosts 83, 84, 115
    - comparisonRules 254, 255
    - definition 205, 229, 232
    - deployment 208, 229
    - deploymentConfiguration 199
    - deployNRun 210, 268
    - dnrDeployment 269, 272
    - dnrDir 269, 270
    - dnrFile 269
    - excludePath 249
    - excludePattern 251
    - execDeploymentTask 89, 209, 232
    - filters 249
    - initiatorProperties element 88
    - listenerProperties 76
    - localNode 83, 203, 280, 283, 288
    - logRules 85, 115, 174
    - nextDeployment 235
    - node 72
    - nodeRef 204, 225, 253
    - nodeSet 71
    - oldOdHome 89
    - path 84, 212, 213, 217
    - pathSpecification 212, 213, 217, 253
    - replicationFarm 204, 231
    - replicationFarmSet 204
    - reverseSource 237
    - reverseTarget 237
    - schedulerProperties 78
    - script 273
    - source 26, 205, 206, 213
    - sourceFilesystem 212, 219, 220, 221, 222
    - sourceTransferRules 264
    - target 26, 205, 207, 213
    - targetFilesystem 219
    - targetRules 223, 225, 253
    - teamsiteProperties 77
    - transferRules 253, 256, 257, 264
    - transportProperties 77
  - encryption 301
    - asymmetric key
      - encryption 282
    - ciphers 288
    - configuration 280

- symmetric key 280
- types 87, 280
- exclude path filter 322
- exclude pattern filter 322
- excludePath element 249
- excludePattern element 251
- execDeploymentTask
  - element 89, 232

**F**

- fan-out deployments 34, 38, 230
  - transactional 232
- file attribute 258
- file comparison rules 254
- file deployment criteria 31
  - comparison-based 31
  - filelist-based 32
- file integrity, checking 146
- file list 35, 219
  - area 223
  - file editing 221
  - file specification 221
  - target host 223
  - TeamSite 222
  - UNIX 221
  - Windows 221
- file permission rules 258
- file transfer rules 256
- filelist attribute 219, 220, 221, 222
- files
  - base server configuration 74
  - base server log 159, 176
  - log 126, 156, 174
  - macro deployment log 162
  - micro deployment log 166
  - receiver configuration 90
  - receiver log 161

- receiver macro deployment
  - log 165
- receiver micro deployment
  - log 169
- service configuration 65
- source macro deployment
  - log 163
- source micro deployment
  - log 167
- filesystem exclusion filtering 249
- filtered deployments 249
  - configurations 251
  - file deletions 253
  - filesystem exclusion 249
  - override precedence 253
  - pattern exclusion 251
  - source-side 252
  - target-side 252
- filters 321
  - deleting 322
  - exclude path 322
  - exclude pattern 322
  - target-side 334
- filters element 249
- firewall authentication 104
- follow links 264
  - source-side 264, 323
  - target-side 264, 323
- followLinks
  - (sourceTransferRules)
    - attribute 264
- followLinks (transferRules)
  - attribute 256, 264
- from attribute 260

**G**

- group attribute 259
- group ownership transferal 260

- group translation 333

## H

- host attribute 72, 203
- host names 71, 72
  - case 73
- hosts
  - adding 122
  - changing 124
  - defining target nodes 70
  - deleting 124
  - log files 126
  - managing 122

## I

- ignoreAcls attribute 254
- ignoreGroup attribute 255
- ignoreModes attribute 254
- ignoreUser attribute 255
- initiatorProperties element 88
- installation 46, 52
  - administration server 46, 53
  - base server 46, 53
  - JDK 1.3 software 92
  - multi-host 51
  - OpenAPI 55
  - opendeploy.war file 48, 95
  - operations server 46, 53
  - receiver 46, 54
  - single-host 49
  - strategies 49
  - target hosts 51
  - Tomcat server 48
  - UNIX 60
  - upgrades 64
  - Windows 56
  - Windows TMP system
    - variable 59

- internationalization 96
  - base server configuration
    - file 96
  - deployment configuration
    - files 96
  - encoding 96
  - nodes configuration file 96
  - receiver configuration file 96
  - service configuration file 96
- invokeOnSuccess attribute 235
- IP address checking 84
- isClearPassword attribute 78, 81
- iwodschedactivate 191
- iwodschedadd 186
- iwodscheddelete 189
- iwodschedget 188
- iwodservergetversion 130
- iwodserverreset 115, 116
- iwodserverstatus 131
- iwodstart 143, 144, 170
- J**
  - jdbcDriverClass attribute 81
  - JDK 1.3 software 92
- L**
  - legacy Web sites, deploying 291
  - level attribute 86, 172
  - listenerProperties element 76
  - localNode element 83, 203, 280, 283, 288
  - location (dnrDeployment)
    - attribute 272
  - location (dnrDir) attribute 270
  - location (dnrFile) attribute 269
  - log files
    - archived 174
    - base server 126
    - location 156
    - macro deployment 162
    - micro deployment 166
    - monitoring 125
    - receiver 126
    - receiver macro
      - deployment 165
    - receiver micro
      - deployment 169
    - recovery 176
    - size 174
    - source macro deployment 163
    - source micro deployment 167
    - viewing 156
  - logging 155
    - base server 159, 173
    - command line 170
    - default server-based 85
    - Deploy and Run 275
    - deployment configurations 228
    - deployment recovery 176
    - file location 156
    - file size 174
    - hierarchy 173
    - host file recovery 176
    - levels 169, 300
    - macro deployment 153, 162, 173
    - micro deployment 166, 173
    - normal 169
    - receiver 161, 173
    - receiver macro
      - deployment 165
    - receiver micro
      - deployment 169
    - rollover naming 174
    - rollover threshold 174, 299
    - rules 299
    - source macro deployment 163
    - source micro deployment 167
    - target host 91
    - user interface 170
    - verbose 169
  - logging levels
    - normal 86, 142, 169, 172, 276, 300
    - verbose 86, 142, 169, 172, 174, 276, 300
  - logical names 71, 72
  - login
    - first time 121
    - normal 120
  - logRules element 85, 115, 174
- M**
  - macro deployment logs 153, 162
    - recovery 176
  - mask (dnrDir) attribute 271
  - mask (dnrFile) attribute 269
  - maxBytes attribute 86, 171, 174
  - micro deployment logs 166
    - recovery 176
  - multi-host installation 51
  - multi-tiered deployments 39, 233
    - nextDeployment element 235
    - transactional 230
- N**
  - name (node) attribute 72
  - name (path) attribute 84
  - name (replicationFarm)
    - attribute 204
  - name (sourceFilesystem)
    - attribute 212
  - nextDeployment element 235
  - next-tier deployments 304

- node element 72
- nodeRef element 204, 225, 253
  - elements
    - nodeRef 231
- nodes
  - naming 71
  - specifying target 71
  - target 70
- nodes attribute 83
- nodes configuration file 70, 128
  - encoding 71, 91, 96
  - uploading 127
- nodeSet element 71
- normal logging level 86, 142, 169, 172, 276, 300
- notation conventions 12

## O

- oldOdHome element 89
- omask attribute 258
- online help 13
- OpenAPI 55, 57, 62, 117, 118
  - bootstrap administrator 68
  - dependencies 117
  - operations 117
  - operations server 47
  - RMI registry service 118
  - starting 111
  - stopping 114
- OpenDeploy
  - ACLs 262
  - administration server 25, 47
  - advantages 23
  - attributes 195, 197
  - base server 24, 48
  - configuration 45
  - configuration DTDs 195
  - daemons 69, 111

- DataDeploy integration 97
- definitions 30
- Deploy and Run 267
- deployment configurations 28
- deployment types 33, 195, 293
- documentation 11, 13
- elements 195
- encryption 87, 280
- environment 24
- file comparison rules 254
- file deployment criteria 31
- file integrity 146
- file permission rules 258
- file transfer rules 256
- host names 71, 72
- hosts 122
- how works 26
- installation 45, 46, 49, 52, 56, 60
- internationalization 96
- introduction to 15
- logging 155
- logical names 71, 72
- login 120, 121
- monitoring 148
- online help 13
- OpenAPI 117
- operations server 25, 46
- receiver 48
- refreshing 115
- roles 42, 132
- schedules 177
- services 69, 110, 113
- software 24
- source host 24, 26
- source/target relationship 26
- starting 109, 111
- status 131
- stopping 113, 115

- target host 25, 26
- TeamSite usage 25
- uninstalling 105
- upgrades 64
- user interface 32, 109, 112, 119
- version 130
- Web site integrity 146
- OpenDeploy Administration Guide*
  - notation conventions 12
  - usage 11
- OpenDeploy Reference* 13
- OpenDeploy Release Notes* 13, 45, 119
- opendeploy.war file 48, 95
- operations server 25, 46
  - bootstrap administrator 68
  - installation 53
  - managing 117
  - OpenAPI 47, 55, 117
  - software 46
  - starting 111
  - stopping 114
  - TeamSite 118
  - uninstallation 107
  - UNIX 47
  - Windows 46
- overrides
  - source-based 223
  - target-based 225

## P

- package files 278
- parameter substitution 265
- path attribute 89
- path element 84, 212, 213, 217
- pathSpecification element 212, 213, 217, 253
- pattern exclusion filtering 251

- pendSession attribute 88
- permission rules 329
- permissions, file 258
- port attribute 72
- ports 72
  - bind port 76
- preserveAcls attribute 256
- previousArea attribute 153, 215, 216, 217, 219, 291

## R

- receiver 48
  - authentication 104
  - installation 46, 54
  - logging 161
  - modifying 65
  - software 51
  - starting 112
  - stopping 115
  - uninstallation 107
  - UNIX 49
  - Windows 49
- receiver configuration file 90, 128
  - allowed directories 92
  - allowed hosts 91
  - buffer size 91
  - encoding 96
  - encryption 92
  - host communication 91
  - logging 91
  - uploading 127
- receiver log file 126, 161
  - recovery 176
- receiver macro deployment log
  - file 165
- receiver micro deployment log
  - file 169
- regex attribute 251
- replication farms 303
- replicationFarm element 204, 231
- replicationFarmSet element 204
- reverse deployments 41, 236
  - symmetric key encryption 281
- reverseSource element 237
- reverseTarget element 237
- revert attribute 254
- RMI registry service 52
  - OpenAPI 118
- rmReadOnly attribute 257
- roles 132
  - Administrator 42, 132, 134, 136
  - management 133
  - server 136, 137
  - system 134
  - User 43, 133, 134, 136, 138, 139
- rollover threshold 299
  - naming 174
  - size 174
- rollover threshold, logging 174
- rules
  - file comparison 254
  - file permission 258
  - file transfer 256

## S

- scenarios, deployment 37
- scheduled deployments 177
  - activating 184, 190
  - command line 185
  - creating 179
  - database configuration 78
  - deactivating 184, 190
  - deleting 184, 189
  - editing 183
  - user interface 178
  - viewing 181
- scheduler database 78
  - In-Memory URL 79
  - Stand-Alone URL 79
  - URL options 79
- schedulerProperties element 78
- script element 273
- scripting, Deploy and Run 273
- server configuration file 280
- server roles 137
  - access 136
- service configuration file 65
  - internationalization 96
- services
  - resetting 115
  - starting 110
  - stopping 113
- setAccess attribute 259
- simulated deployments 145
- single-host installation 49
- source area
  - specifying location 212, 217
- source element 26, 205, 206, 213
- source file location 206, 316, 320
- source hosts 26
  - area 212, 215
  - defined 24
  - directory comparison 212
  - installation 46
  - overrides 223
  - TeamSite comparison 215
- source macro deployment log
  - file 163
- source micro deployment log
  - file 167
- source-based overrides 223
- sourceFilesystem element 212, 219, 220, 221, 222

- sourceTransferRules
  - element 264
- sslCaCertificate attribute 287
- sslCertificate attribute 287
- sslCiphers attribute 288
- sslPrivateKey attribute 287
- sslVerifyPeer attribute 287
- state (dnrDeployment)
  - attribute 272
- state (dnrDir) attribute 271
- state (dnrFile) attribute 269
- stdin program 275
- stdout program 275
- subPath attribute 249
- svrTryCount attribute 257
- svrTryDisableOverwrite
  - attribute 257
- svrTryInterval attribute 257
- symmetric key encryption 280
  - configuration 280
  - reverse deployments 281
- system role access 134

## T

- target element 26, 205, 207, 213
- target file location 207, 325
  - alternate 223
- target hosts 26
  - alternate 223
  - area 213, 218, 223, 227
  - defined 25
  - directory comparison 213
  - features, defining 227
  - file list 223
  - filesystem-based 219
  - installation 46
  - mixed platform 225
  - overrides 225

- TeamSite comparison 218
- target replication farms 203
- target-based overrides 225
- targetFilesystem element 219
- targetRules element 223, 225, 253
- TeamSite
  - OpenAPI 55
  - operations server 118
  - using 25
- TeamSite comparison 31, 214
  - area 215, 218
  - Deploy and Run scripts 219
  - deployments 34
  - legacy Web sites 291
  - previousArea 216
  - source host 215
  - target host 218
- teamsiteProperties element 77
- test deployments 145
- timeout setting 121
- to attribute 261
- Tomcat server
  - configuration file 92, 93, 94
  - installation 48
- transactional attribute 209
- transactional deployments 40,
  - 208, 229, 305
  - multi-tiered 230
- transferRules element 253, 256,
  - 257, 264
- transmission rules 328
- transportProperties element 77

## U

- uninstallation 105
  - administration server 107
  - base server 107
  - operations server 107

- receiver 107
- UNIX 106
- Windows 105
- upgrades, OpenDeploy 64
- useDefinition attribute 232
- useNode attribute 204, 231
- user attribute 259
- user interface 32, 119
  - browser requirements 119
  - hosts 122
  - scheduled deployments 178
  - starting 112, 119
  - timeout setting 121
- user ownership transferal 260
- User role 43, 133, 134, 136, 139
  - access 138
- user translation 332
- useReplicationFarm
  - (reverseSource)
    - attribute 237
- useReplicationFarm (target)
  - attribute 207

## V

- verbose logging level 86, 142,
  - 169, 172, 174, 276, 300
- version attribute 77

## W

- Web site integrity, checking 146
- when (dnrDeployment)
  - attribute 272
- when (dnrDir) attribute 270
- when (dnrFile) attribute 269
- where attribute 273
- Windows TMP system
  - variable 59

**X**

XML code 147